FINAL PROGRESS REPORT

METAL-MATRIX COMPOSITES AND POROUS MATERIALS: CONSTITUTIVE MODELS, MICROSTRUCTURE EVOLUTION AND APPLICATIONS

AFOSR GRANT F49620-97-1-0212

PERIOD FROM 5/1/97 TO 12/31/99

TOTAL AMOUNT: $59,782

P. Ponte Castañeda

Department of Mechanical Engineering and Applied Mechanics
University of Pennsylvania
Philadelphia, PA 19104-6315

REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-00-

0134

1. REPORT DATE *(DD-MM-YYYY)*: 23-02-2000

2. REPORT TYPE: Final

3. DATES COVERED *(From - To)*: 01-05-1997 to 31-12-1999

4. TITLE AND SUBTITLE

METAL-MATRIX COMPOSITES AND POROUS MATERIALS: CONSTITUTIVE MODELS, MICROSTRUCTURE EVOLUTION AND APPLICATIONS

5a. CONTRACT NUMBER

5b. GRANT NUMBER: F49620-97-1-0212

5c. PROGRAM ELEMENT NUMBER

5d. PROJECT NUMBER: 2304

5e. TASK NUMBER: BX

5f. WORK UNIT NUMBER

6. AUTHOR(S)

P. Ponte Castañeda

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Trustees of the University of Pennsylvania
133 S. 36th Street, Mezzanine
Philadelphia, PA 19104

8. PERFORMING ORGANIZATION REPORT NUMBER: 530859

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Air Force Office of Scientific Research/NM
110 Duncan Avenue, Room B115
Bolling AFB DC 20332-8050

10. SPONSOR/MONITOR'S ACRONYM(S): AFOSR/NM

11. SPONSOR/MONITOR'S REPORT NUMBER(S): FQ8671-9900130

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release, distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Constitutive models were developed and implemented numerically to account for the evolution of microstructure and anisotropy in finite-deformation processes involving porous and composite materials. Use was made of rigorous homogenization procedures developed by the PI under an earlier grant (AFOSR 89-0288). The constitutive models developed took a standard internal- variable form depending on suitably derived internal variables (serving to characterize the current state of the microstructure) and differential evolution laws for these variables. The main findings of the work are: (a) the evolution of the microstructure affects quite significantly the macroscopic response of porous materials; (b) there is synergistic coupling between the various micro- structural variables, in particular, between the porosity and anisotropy; (c) qualitative agreement has been found with available experimental results, particularly for the non-uniform evolution of porosity and anisotropy in forming processes. The particular case of porous metals was considered in detail and the newly developed constitutive models were implemented in a general- purpose, finite-strain finite element program. Several problems in the area of metal forming were then analyzed, and implications for the onset of shear instabilities was explored.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
U	U	U	UU		P. Ponte Castañeda
					19b. TELEPHONE NUMBER *(Include area code)*: 215-898-5046

1. Grant Number

AFOSR F49620-97-1-0212

2. Objectives

The objective of this work is to develop constitutive models that are capable of accounting for the evolution of microstructure in finite-deformation processes of heterogeneous materials. The models to be developed, through rigorous homogenization procedures for random microstructures, will take the form of standard homogenized constitutive models, which will depend explicitly on appropriate internal variables (serving to characterize the current state of the microstructure), and will be supplemented by differential evolution laws for these internal variables. The particular case of a porous metal will be considered in some detail and the newly developed constitutive models will be implemented in a general-purpose, finite-strain finite element program. Several problems in the areas of metal forming will then be analyzed, and implications for the onset of shear instabilities will be explored. Finally, comparisons with experimental results and further refinements of the models will be carried out, for porous aluminum and aluminum-matrix composites.

3. Status of effort

This project has been completed as of 12/31/99.

3. Accomplishments

The developed constitutive models are the first of their type to be able to account for deformation-induced anisotropy in porous materials and particle-reinforced composites. The main findings of the work are: (a) the evolution of the microstructure affects quite significantly the macroscopic response of porous materials; (b) there significant coupling between the various microstructural variables, in particular, between the porosity and anisotropy; (c) qualitative agreement has been found with available experimental results, particularly for the evolution of porosity and anisotropy in forming processes. A constitutive subroutine (UMAT) for the model has been successfully implemented in ABAQUS and tested for several different types of forming processes, including plane-strain and axisymmetric extrusion and tapered-disk compaction. In the context of future work, we will explore further the implications of microstructure evolution on the overall stability of deformation processes and consider applications to problems where microstructure evolution is significant, including ductile fracture, high-strain rate (explosive loading) applications and impact loading.

5. Personnel

The PI was supported for one month/year of summer salary for his efforts on this project. In addition, one former graduate student, M. Kailasam, now employed by HKS (the makers of ABAQUS), assisted with the computations on his free time. In addition, Professor Nick Aravas, now at the University of Thessaly (Greece) collaborated with us on the numerical implementation of the constitutive model and in the writing of the papers.

6. Publications

Sponsored by AFOSR

M. Kailasam, N. Aravas and P. Ponte Castañeda. "Constitutive models for porous materials with developing anisotropy and application to deformation processing." *Computational Mechanics* (1999): to appear.

N. Aravas, M. Kailasam and P. Ponte Castañeda. "Constitutive Models for Porous Media with Microstructure Evolution: Computational Issues." *Proceedings of the 3rd National Congress on Computational Mechanics,* Volos (Greece), pp. 65-72, 1999.

P. Ponte Castañeda and J. R. Willis. "Variational second-order estimates for nonlinear composites." *Proceedings of the Royal Society of London A* **455**(1999): 1799-1812.

M. Kailasam and P. Ponte Castañeda. "A general constitutive theory for linear and nonlinear particulate media with evolving microstructure." *Journal of the Mechanics and Physics of Solids* **46** (1998): 427-465.

M. Nebozhyn and P. Ponte Castañeda. "Exact second-order estimates of the self-consistent type for nonlinear composite materials." *Mechanics of Materials* **28** (1998): 9-22.

Related work sponsored by other agencies.

M. Nebozhyn, P. Gilormini and P. Ponte Castañeda. "Variational self-consistent estimates for cubic viscoplastic polycrystals: The effects of grain anisotropy and shape." *Journal of the Mechanics and Physics of Solids* (2000): submitted for publication.

M. Nebozhyn, P. Gilormini and P. Ponte Castañeda. "Variational self-consistent estimates for viscoplastic polycrystals with highly anisotropic grains." *Comptes Rendus de l'Académie des Sciences, Paris, Série IIb* (2000): communicated.

P. Ponte Castañeda and E. Tiberio. "A second-order homogenization procedure in finite elasticity and applications to black-filled elastomers." *Journal of the Mechanics and Physics of Solids* (2000): in press.

M. Nebozhyn and P. Ponte Castañeda. "The second-order procedure: Exact versus approximate results for isotropic, two-phase composites." *Journal of the Mechanics and Physics of Solids* **47** (1999): 2171-2185..

P. Ponte Castañeda and E. Tiberio. "Homogenization estimates for hyperelastic composites and applications to particle-reinforced rubbers." *Comptes Rendus de l'Académie des Sciences, Paris, Série IIb* **327** (1999): 1297-1304.

K. Bose, P. A. Mataga and P. Ponte Castañeda. "Stable crack growth along a brittle/ductile interface. interface - II. Small scale yielding solutions and interfacial toughness predictions." *International Journal of Solids and Structures* **36** (1998): 1-34.

M. Bornert and P. Ponte Castañeda. "Second-order estimates of the self-consistent type for viscoplastic polycrystals." *Proceedings of the Royal Society of London A* **454** (1998): 3035-3045

P. Ponte Castañeda and P. Suquet. "Nonlinear composites." *Advances in Applied Mechanics* **34** (1998): 171-302.

P. Ponte Castañeda. "Three-point bounds and other estimates for strongly nonlinear composites." *Physical Review B* **57** (1998): 12077-12083.

M. Nebozhyn and P. Ponte Castañeda. "Second-order estimates for the effective behavior of nonlinear porous materials." In *Transformation Problems in Composite and Active Materials* (*IUTAM* Symposium), edited by Y. Bahei-El-Din and G. J. Dvorak, 73-88. New York: Kluwer, 1998.

M. Nebozhyn and P. Ponte Castañeda. "Variational estimates of the self-consistent type for creep of polycrystalline materials." In *Micro- and Macrostructural Aspects of Thermoplasticity* (*IUTAM* Symposium), edited by O. T. Bruhns and E. Stein, 207-215. New York: Kluwer, 1998.

P. Ponte Castañeda. "Nonlinear polycrystals with crystallographic and morphological texture evolution." In *Continuum Models and Discrete Systems (CMDS 9)*, edited by E. Inan and K. Z. Markov, 228-235. Singapore: World Scientific, 1998.

Ph.D. Thesis

M. Kailasam (1999) A general constitutive theory for particulate composites and porous materials with microstructure evolution. University of Pennsylvania.

Books Edited

P. Ponte Castañeda and P. Suquet (Guest Eds.), The John R. Willis 60th Anniversary Issue, *Journal of the Mechanics and Physics of Solids* **48** (2000): 400 pages.

7. Interactions

Technical Committees

ASME Technical Committee on Instability in Solids and Structures, Member.

Seminars

"Microstructure evolution in porous and particle-reinforced composites."
Laboratoire de Mécanique des Solides, *Ecole Polytechnique* (France), June 22, 1998.

"Nonlinear homogenization and applications."
Department of Materials Science,*UniversidadPolitecnica de Madrid,* June 28, 1999.

"Homogenization estimates in finite elasticity and applications to particle-reinforced elastomers." *Isaac Newton Institute for Mathematical Sciences,* Cambridge University, September 12, 1999.

"Microstructure evolution in porous and composite materials: Implications for shear localization." *Isaac Newton Institute for Mathematical Sciences,* Cambridge University, October, 1999.

Invited Conference Presentations

"Micromechanics of Nonlinear Composites."
9th International Symposium on Continuum Models and Discrete Systems,
Istambul (Turkey), June 29-July 3, 1998.

"Microstructure evolution in particle-reinforced systems."
7th International Symposium on Plasticity and its Current Applications,
Cancun (Mexico), January 5-13, 1999.

"Nonlinear Homogenization and Applications."
7th International Symposium on Plasticity,
Cancun (Mexico), January 5-13, 1999.

"Constitutive Models for Porous Media with Microstructure Evolution and Application to Forming." *3rd National Congress on Computational Mechanics,*
Volos (Greece), June 24-26, 1999.

Transitions

Re-initiated contact with Dr. Richard Becker, formerly at ALCOA, and now at Lawrence Livermore. Dr. Becker has proposed the implementation of our anisotropic constitutive model for porous metals into the computer programs at LLNL, as part of the ASCI initiative. If the project is approved, the constitutive model will be extended for the materials of interest at LLNL and implemented numerically.

8. Honors and Awards

Associate Editor, 1999-.
Comptes Rendus de l'Académie des Sciences de Paris, Série IIb, Mécanique.

Two lectures at the Isaac Newton Institute Program on Mathematical Developments in Solid Mechanics and Materials Science, University of Cambridge, September-December 1999.

Invited to participate and lecture in the Workshop on "Homogenization and Effective Media Theories." *Mathematical Sciences Research Institute,* Berkeley, March 6-17, 2000.

Invited to give a "Sectional Lecture" on "Nonlinear Composites" at the upcoming 20th International Congress of Theoretical and Applied Mechanics, August 2000.

9. Appendix: The finite element program

N.f

```
c
C----- UMAT routine
      subroutine umat(stress,statev,ddsdde,sse,spd,scd,
     &                  rpl,ddsddt,drplde,drpldt,
     & stran,dstran,time,dtime,temp,dtemp,predef,dpred,cmname,
     & ndi,nshr,ntens,nstatv,props,nprops,coords,drot,pnewdt,
     & celent, dfgrd0,dfgrd1,noel,npt,layer,kspt,kstep,kinc)

c     include 'ABA_PARAM.INC'
      character*8 cmname
      real*8 stress(ntens),statev(nstatv),ddsdde(ntens,ntens)
      real*8 ddsddt(ntens),drplde(ntens),stran(ntens),dstran(ntens)
      real*8 time(2),predef(1),dpred(1),props(nprops),coords(3)
      real*8 drot(3,3),dfgrd0(3,3),dfgrd1(3,3)
      real*8 sse,spd,scd,rpl,drpldt,dtime,temp,dtemp
      real*8 pnewdt,celent,ct,st,cp,sp,cs,ss
      integer ntens,ndi,nshr,nstatv,nprops,noel,npt,layer
      integer kspt,kstep,kinc,i,j,inpt
c
      real*8 mu1e,k1e,mu2e,ek2e
      real*8 s(3,3),strain(3,3),ddsdde_mine(6,6)
      real*8 f,wi1,wi2,dlam,atheta,aphi,apsi
      real*8 sc(6),strainc(6),dummy3(3,3)
      real*8 dummy1(3,3),R(3,3),RT(3,3),dummy2(3,3)
      real*8 mu1,k1,mu2,k2,sigy1,x2,xacc,sigy0,x1,hrd
      real*8 eps_plastic,k11,rtsafe,rot(3,3),rott(3,3)
      real*8 rtnewt,eigen(3,3),ei2(3),ei3(3),ei1(3)
      real*8 a,b,c,xb1,xb2
      real*8 func
      real*8 eigval(3)
      real*8 rota(3,3),rotj(3,3)
      real*8 tottheta,totphi,totpsi
      integer nb,num,n,np,nrot
      logical path,yield,check,neg,debug
      logical evolf,evolwi,evolti
      logical zb,cal
      common /controls/evolf,evolwi,evolti
      common /mkmodulus/mu1,k1,mu2,k2,sigy0,k11
      common /mkelas/mu1e,k1e,mu2e,ek2e
      common /mkdata1/f,wi1,wi2,sc,strainc,eps_plastic,sigy1,hrd
      common /mkorient/cp,sp,ct,st,cs,ss,rot,rott,atheta,aphi,apsi
      common /mkdata2/yield,check,neg
      common /mkrot/R,RT,rotj
      common /mdebug/debug
      common /cal1/cal
      common /zb1/zb
      external func,rtsafe,funcd,rtnewt,zbrak
      path=.true.
     open(unit=15,file='/scratch/mahesh/data',
     &status='unknown')
     open(unit=16,file='/scratch/mahesh/data1',
     &status='unknown')
c
c---- Various parameters, like those used to turn the evolution of
c     some of the micro. variables on/off etc. are entered below.
c     if (((noel.eq.150).or. (noel.eq.600)).and.(npt.eq.1)) then
      write(15,*)'kinc',kinc,'npt',npt,'ntens',ntens
      write(15,*)'stress'
      write(15,*)stress
c     cal=.true.
      call flush(15)
c     else
c      cal=.false.
c     endif
      evolf=.true.
      evolwi=.true.
      evolti=.true.
      debug=.false.
      if(debug.eq..true.) then
      write(15,*)'entry'
      call flush(15)
      endif
      inpt=npt
      yield=.false.
      neg=.false.
      mu1=props(1)
      k1=props(2)
      mu2=props(3)
      k2=props(4)
      sigy0=props(5)
      mu1e=props(6)
      k1e=props(7)
      mu2e=props(8)
      ek2e=props(9)
```

```
      k11=k1*1.0
      if(debug.eq..true.) then
      write(15,*)'elas',mu1e,k1e,mu2e,ek2e
      call flush(15)
      endif
      do 1 i=1,3
       do 2 j=1,3
        if (dabs(dfgrd0(i,j)).lt.1.0d-14) then
       dfgrd0(i,j)=0.0
        endif
        if (dabs(dfgrd1(i,j)).lt.1.0d-14) then
         dfgrd1(i,j)=0.0
     endif
2      continue
1     continue
      do 3 i=1,6
     if (dabs(stress(i)).lt.1.0d-08) then
      stress(i)=0.0
       endif
3     continue
c
c------ Convert stress to column form in MY notation
c------ The input comes in the form: {s11,s22,s33,s12,s13,s23}
c       while I have always used
c         {s11,s22,s33,sqrt(2.0)*s12,sqrt(2.0)*s23,sqrt(2.0)*s31}.
c       Below I convert the incoming stresses to MY notation
c------ Also note that in the case of use with plane-strain elements
c       only 4 components of the stress are provided by ABAQUS and these
c       are the 11,22,33 and 12 components. In my notation, plane calculations
c       are performed in the 2-3 plane. So care has to be taken
c       in converting to my notation.
      if (ntens.eq.4) then
       sc(1)=stress(3)
       sc(2)=stress(1)
       sc(3)=stress(2)
       sc(4)=0.0
       sc(5)=dsqrt(2.D0)*stress(4)
       sc(6)=0.0
      else
       sc(1)=stress(1)
       sc(2)=stress(2)
       sc(3)=stress(3)
       sc(4)=dsqrt(2.D0)*stress(4)
       sc(5)=dsqrt(2.D0)*stress(6)
       sc(6)=dsqrt(2.D0)*stress(5)
      endif
      s(1,1)=sc(1)
      s(2,2)=sc(2)
      s(3,3)=sc(3)
      s(1,2)=sc(4)/dsqrt(2.D0)
      s(2,3)=sc(5)/dsqrt(2.D0)
      s(3,1)=sc(6)/dsqrt(2.D0)
      s(2,1)=s(1,2)
      s(3,2)=s(2,3)
      s(1,3)=s(3,1)
c------ Decompose deltaF to get R and U
c       Again, we must convert dfgrd0 to my notation:
       if (ntens.eq.4) then
      dummy1(1,1)=dfgrd0(3,3)
      dummy1(2,2)=dfgrd0(1,1)
      dummy1(3,3)=dfgrd0(2,2)
      dummy1(1,2)=dfgrd0(3,1)
      dummy1(2,1)=dfgrd0(1,3)
      dummy1(2,3)=dfgrd0(1,2)
      dummy1(3,2)=dfgrd0(2,1)
      dummy1(1,3)=dfgrd0(3,2)
      dummy1(3,1)=dfgrd0(2,3)
       else
        do 10 i=1,3
         do 20 j=1,3
          dummy1(i,j)=dfgrd0(i,j)
20       continue
10      continue
       endif
       call inverse3x3(dummy1,dummy2)
       if (ntens.eq.4) then
      dummy1(1,1)=dfgrd1(3,3)
      dummy1(2,2)=dfgrd1(1,1)
      dummy1(3,3)=dfgrd1(2,2)
      dummy1(1,2)=dfgrd1(3,1)
      dummy1(2,1)=dfgrd1(1,3)
      dummy1(2,3)=dfgrd1(1,2)
      dummy1(3,2)=dfgrd1(2,1)
      dummy1(1,3)=dfgrd1(3,2)
```

```
        dummy1(3,1)=dfgrd1(2,3)
         else
          do 11 i=1,3
           do 21 j=1,3
            dummy1(i,j)=dfgrd1(i,j)
21         continue
11        continue
         endif
         call matprod(dummy1,dummy2,dummy3)
         call decompose(dummy3,R,strain,eigen,eigval)
c        write(16,*)'dfgrd11',dfgrd1(1,1),dfgrd1(1,2),dfgrd1(1,3)
c        write(16,*)'dfgrd12',dfgrd1(2,1),dfgrd1(2,2),dfgrd1(2,3)
c        write(16,*)'dfgrd13',dfgrd1(3,1),dfgrd1(3,2),dfgrd1(3,3)
c        write(16,*)'strain1',strain(1,1),strain(1,2),strain(1,3)
c        write(16,*)'strain2',strain(2,1),strain(2,2),strain(2,3)
c        write(16,*)'strain3',strain(3,1),strain(3,2),strain(3,3)
c------ Here we have obtained ln(delta_U) and R, the components of which are
c        relative to the fixed Lab coords.
c        write(15,*)'eigen',eigen
c        call flush(15)
         ei1(1)=eigen(1,1)
         ei1(2)=eigen(1,2)
         ei1(3)=eigen(1,3)
       ei2(1)=eigen(1,2)
       ei2(2)=eigen(2,2)
       ei2(3)=eigen(3,2)
       ei3(1)=eigen(1,3)
       ei3(2)=eigen(2,3)
       ei3(3)=eigen(3,3)
cwrite(15,*)'test1=',ei1(1)*ei2(1)+ei1(2)*ei2(2)+ei1(3)*ei2(3)
cwrite(15,*)'test2=',ei1(1)*ei3(1)+ei1(2)*ei3(2)+ei1(3)*ei3(3)
cwrite(15,*)'test3=',ei2(1)*ei3(1)+ei2(2)*ei3(2)+ei2(3)*ei3(3)
cwrite(15,*)'test4=',ei1(1)*ei1(1)+ei1(2)*ei1(2)+ei1(3)*ei1(3)
cwrite(15,*)'test5=',ei2(1)*ei2(1)+ei2(2)*ei2(2)+ei2(3)*ei2(3)
cwrite(15,*)'test6=',ei3(1)*ei3(1)+ei3(2)*ei3(2)+ei3(3)*ei3(3)
c        write(15,*)'t7',eigval(1),'    ',ei1(1),ei1(2),ei1(3)
c        write(15,*)'t8',eigval(2),'    ',ei2(1),ei2(2),ei2(3)
c        write(15,*)'t9',eigval(3),'    ',ei3(1),ei3(2),ei3(3)
         if ((kinc.eq.1).and.(kstep.eq.1)) then
          f=0.15
          wi1=1.0001
          wi2=1.0001
        atheta=0.0
          aphi=0.0
          apsi=0.0
        tottheta=0.0
        totphi=0.0
        totpsi=0.0
        do 777 i=1,3
         do 778 j=1,3
          if (i.ne.j) then
           rotj(i,j)=0.0
            else
           rotj(i,j)=1.0
            endif
778        continue
777       continue
          statev(1)=f
          sigy1=props(5)
          statev(2)=wi1
          statev(3)=wi2
          statev(4)=atheta
          statev(5)=aphi
          statev(6)=apsi
          statev(7)=0.0
          statev(8)=props(5)
        statev(9)=0.0
        statev(10)=hrd
        statev(11)=wi1/wi2
        statev(12)=rotj(1,1)
        statev(13)=rotj(1,2)
        statev(14)=rotj(1,3)
        statev(15)=rotj(2,1)
        statev(16)=rotj(2,2)
        statev(17)=rotj(2,3)
        statev(18)=rotj(3,1)
        statev(19)=rotj(3,2)
        statev(20)=rotj(3,3)
        statev(21)=tottheta
        statev(22)=totphi
        statev(23)=totpsi
        statev(24)=1.0/wi2
        statev(25)=(1.0/wi2)*wi1
         endif
```

```fortran
      do 5 i=1,3
       do 6 j=1,3
        if (dabs(R(i,j)).lt.1.0d-14) then
         R(i,j)=0.0
        endif
6       continue
5      continue
       do 100 i=1,3
        do 110 j=i,3
         RT(i,j)=R(j,i)
110     continue
100    continue
c----- BEGIN TEST1
c----- This is a test to see if the components of [drot][s]_t[drot_transpose]
c      that ABAQUS gives me are with respect to the fixed frame. So, what I
c      do below is assume that this is true (they are with respect to the
c      GLOBAL axes) and then use transformation laws to obtain the components
c      w.r.t to a frame which is obatined from the GLOBAL frame by a rotation
c      of drot. These components are the same as the components of [s]_t w.r.t
c      the GLOBAL frame. I then check if these values agree with the stress
c      of the previous increment: this is output in the .dat file and since
c      all tensors are stored w.r.t the GLOBAL axes, the stress output should
c      agree with 'dummy2' below and they do!
c      write(15,*)'stress_1'
c      dummy2(1,1)=stress(1)
c      dummy2(2,2)=stress(2)
c      dummy2(3,3)=stress(3)
c      dummy2(1,2)=stress(4)
c      dummy2(2,1)=stress(4)
c      dummy2(2,3)=stress(6)
c      dummy2(3,2)=stress(6)
c      dummy2(3,1)=stress(5)
c      dummy2(1,3)=stress(5)
c      call matprod(RT,dummy2,dummy1)
c      call matprod(dummy1,R,dummy2)
c      write(15,*)dummy2
c----- END TEST1
c----- BEGIN TEST2
c----- From dfgrd0 and dfgrd1, I have obtained ln(delta_U) and R. R is exactly
c      the same as drot. The dstran that ABAQUS provides me is ln(delta_V) and
c      its components are w.r.t the GLOBAL axes. I have the components of
c      ln(delta_U) also w.r.t the GLOBAL axes. It can be easily checked to
c      see that they have the same eigen values. Also [R][ln(delta_U)][RT]
c      must give us dstran.
c      write(15,*)'R'
c      write(15,*)R
c      write(15,*)'drot'
c      write(15,*)drot
c      do 31 i=1,3
c       do 32 j=1,3
c        dummy1(i,j)=strain(i,j)
c32      continue
c31     continue
c      n=3
c      np=3
c      call jacobi(dummy1,n,np,eigval,dummy2,nrot)
c      write(15,*)'Eigen values of ln(delta_U)'
c      write(15,*)eigval
c      do 33 i=1,3
c       do 34 j=1,3
c        dummy1(i,j)=strain(i,j)
c34      continue
c33     continue
c      write(15,*)'ln(delta_U)'
c      write(15,*)dummy1(1,1),dummy1(2,2),dummy1(3,3),dummy1(1,2)*2.0,
c     &dummy1(3,1)*2.0,dummy1(2,3)*2.0
c      call matprod(RT,dummy1,dummy2)
c      call matprod(dummy2,R,dummy1)
c      write(15,*)'ln(delta_V) from ln(delta_U)'
c      write(15,*)dummy1(1,1),dummy1(2,2),dummy1(3,3),dummy1(1,2)*2.0,
c     &dummy1(3,1)*2.0,dummy1(2,3)*2.0
c      write(15,*)'ln(delta_V)'
c      write(15,*)dstran
c----- END TEST2
c------ Stress components that abaqus provides me are w.r.t GLOBAL
c       co-ordinate frame. Since this is a finite-deformation problem,
c       the stress that I have here is one that has been rotated by 'drot',
c       i.e, it is [drot] [stress]_t [drot_transpose], but the components
c       are w.r.t the fixed GLOBAL frame. The integration algorithm that
c       I use requires that I used [stress]_t and NOT the above quantity.
c       This is done below:
c
       call matprod(RT,s,dummy1)
       call matprod(dummy1,R,s)
```

```
c
c-------Note that R and drot are exactly the same. The components of s are
c       still relative to the global frame.
c
c       An important interpretation of why we use the stress in the GLOBAL
c       coordinate frame is that when we add to this the change in stress
c       we get the new stress components relative to a coordinate frame
c       which is obtained from the global frame by a rotation of drot. This
c       stress is called sig_hat_n+1 in Dr. Aravas's notes. We see that to
c       recover stress_n+1, we must pre and post multiply by R and RT,
c       respectively. This is nothing but expressing the new stress components
c       relative the GLOBAL frame.
c
c       Note that we need not use the GLOBAL frame as one where all components
c       are stored. In fact, we find it convenient to perform our integration
c       relative to a coordinate frame which coincides with the orientation
c       of the particles. What this implies is that, we transform the stress
c       we have obtained above to this (particle orientation frame) coordinate
c       frame and perform our integration there. The stress components that we
c       then obtain are then w.r.t a coordinate frame which has is obtained
c       from the particle orientation frame by a rotation of drot. We then
c       again transform the stress (and other variables) to be expressed
c       relative to the GLOBAl frame.
c
        sc(1)=s(1,1)
        sc(2)=s(2,2)
        sc(3)=s(3,3)
        sc(4)=dsqrt(2.D0)*s(1,2)
        sc(5)=dsqrt(2.D0)*s(2,3)
        sc(6)=dsqrt(2.D0)*s(3,1)
       if ((kinc.gt.1).or.(kstep.gt.1)) then
        f=statev(1)
        wi1=statev(2)
        wi2=statev(3)
      c=1.000
      a=c/wi1
      b=c/wi2
        atheta=statev(4)
        aphi=statev(5)
        apsi=statev(6)
        eps_plastic=statev(7)
        sigy1=statev(8)
      rotj(1,1)=statev(12)
      rotj(1,2)=statev(13)
      rotj(1,3)=statev(14)
      rotj(2,1)=statev(15)
      rotj(2,2)=statev(16)
      rotj(2,3)=statev(17)
      rotj(3,1)=statev(18)
      rotj(3,2)=statev(19)
      rotj(3,3)=statev(20)
      tottheta=statev(21)*3.14159/180.0
      totphi=statev(22)*3.14159/180.0
      totpsi=statev(23)*3.14159/180.0
      if (f.lt.0.001) then
       evolf=.false.
       evolwi=.false.
        endif
      if (wi2.lt.0.02) then
       evolwi=.false.
        endif
       endif
c------ The increment of strain below (corresponds to ln(delta_U)) is w.r.t
c       GLOBAL coordinate frame.
        strainc(1)=strain(1,1)
        strainc(2)=strain(2,2)
        strainc(3)=strain(3,3)
        strainc(4)=strain(1,2)*dsqrt(2.D0)
        strainc(5)=strain(2,3)*dsqrt(2.D0)
        strainc(6)=strain(3,1)*dsqrt(2.D0)
c
c------ Expressing stress and strain in local coordinates:
c       The integration problem is carried out in a co-ordinate frame
c       which coincides with the orientation of the particles. This is
c       done because it is more eonomical to rotate the stress and the
c       strain once rather than having to rotate 4th order tensors like
c       Amat, Bmat, MHS etc. whose components are readily obtained w.r.t
c       the co-ordinate frame which coincides with the particle orientation.
c
99      ct=dcos(atheta)
        st=dsin(atheta)
        cp=dcos(aphi)
        cs=dcos(apsi)
        sp=dsin(aphi)
```

```
        ss=dsin(apsi)
        rota(1,1)=cs*cp-ss*ct*sp
        rota(1,2)=-cs*sp-ss*ct*cp
        rota(1,3)=ss*st
        rota(2,1)=ss*cp+cs*ct*sp
        rota(2,2)=-ss*sp+cs*ct*cp
        rota(2,3)=-cs*st
        rota(3,1)=st*sp
        rota(3,2)=st*cp
        rota(3,3)=ct
      call matprod(rota,rotj,rot)
        do 7 i=1,3
         do 8 j=1,3
          rott(i,j)=rot(j,i)
8        continue
7       continue
c------ Below the components of the stress are expressed w.r.t. the
c       a coordinate system which instantaneously coincides with the
c       orientation of the particles.
         call matprod(rott,s,dummy1)
         call matprod(dummy1,rot,s)
         sc(1)=s(1,1)
         sc(2)=s(2,2)
         sc(3)=s(3,3)
         sc(4)=dsqrt(2.D0)*s(1,2)
         sc(5)=dsqrt(2.D0)*s(2,3)
         sc(6)=dsqrt(2.D0)*s(3,1)
c------ Rotating below to express strain w.r.t the orientation of particles.
         call matprod(rott,strain,dummy1)
         call matprod(dummy1,rot,strain)
         strainc(1)=strain(1,1)
         strainc(2)=strain(2,2)
         strainc(3)=strain(3,3)
         strainc(4)=strain(1,2)*dsqrt(2.D0)
         strainc(5)=strain(2,3)*dsqrt(2.D0)
         strainc(6)=strain(3,1)*dsqrt(2.D0)
c------ Integrating to obtain the stress and the new state variables.
        xacc=1.0d-9
        if ((strainc(1).eq.0.0).and.(strainc(2).eq.0.0)
     & .and.(strainc(3).eq.0.0).and.(strainc(4).eq.0.0)
     & .and.(strainc(5).eq.0.0).and.(strainc(6).eq.0.0)) then
         dlam=0.0
         go to 4
        endif
        call guessmaker(x2)
        if (yield.eq..false.) then
         dlam=0.0
         go to 4
        endif
        if (neg.eq..true.) then
         pnewdt=0.75
         write(15,*)'This should never happen!'
       call flush(15)
         return
        endif
        if (x2.lt.0.0) then
c        write(15,*)'Guessmaker screwed up',x2
c        call flush(15)
        endif
c------ Trying to bracket a solution for delta_lambda
        x2=0.02
        x1=1.0d-10
        num=20
        nb=1
        call zbrak(func,x1,x2,num,xb1,xb2,nb)
        if (neg.eq..true.) then
         pnewdt=0.75
       return
        endif
        if (nb.eq.0) then
c        write(15,*)'bracketing problem: Level 1'
       call flush(15)
         x1=1.0d-10
         num=50
         nb=1
         call zbrak(func,x1,x2,num,xb1,xb2,nb)
         if (neg.eq..true.) then
          pnewdt=0.75
          return
         endif
        endif
        if (nb.eq.0) then
c        write(15,*)'bracketing problem: Level 2'
       call flush(15)
```

```
      zb=.true.
        x1=1.0d-10
        num=200
        nb=1
        call zbrak(func,x1,x2,num,xb1,xb2,nb)
        if (neg.eq..true.) then
         pnewdt=0.75
         return
        endif
       endif
       if (nb.eq.0) then
        write(15,*)'bracketing problem: Level critical'
      call flush(15)
        x1=1.0d-10
        num=500
        nb=1
        call zbrak(func,x1,x2,num,xb1,xb2,nb)
        if (neg.eq..true.) then
         pnewdt=0.75
         return
        endif
       endif
      zb=.false.
       if (nb.eq.0) then
        pnewdt=0.75
      write(15,*)'Bracketing Failure'
      call flush(15)
      return
       endif
c----- Using a combined secant/bisection procedure to find the solution for
c      delta_lambda.
c
      dlam=rtsafe(funcd,xb1,xb2,xacc)
c     write(15,*)'dlam',dlam,'kinc',kinc
      if (neg.eq..true.) then
       pnewdt=0.75
       write(15,*)'returned after unsuccessful rtsafe',kinc,noel,npt
      call flush(15)
      return
       endif
c----- Initializing theta: The voids start evolving at an orientation given
c      by the eigen values of the right-stretch tensor. The orientation of the
c      voids makes sense only when the voids are not spherical anymore. This
c      means that the following steps are carried out only when the material
c      has started deforming plastically.
      if (evolti.eq..true.) then
       if (statev(9).eq.0.0) then
        write(15,*)'ei2',ei2
        write(15,*)'ei3',ei3
cwrite(15,*)'thetaj',dacos(rotj(3,3))
        call flush(15)
        if (dabs(ei2(3)).gt.1.0d-10) then
       if (eigval(3).gt.eigval(2)) then
          if ((ei2(3).lt.0.0).and.(ei2(2).gt.0.0)) then
           atheta=-dacos(ei2(2))
         write(15,*) 'theta_initializing_a',noel,npt
          else if ((ei2(3).gt.0.0).and.(ei2(2).gt.0.0)) then
           atheta=dacos(ei2(2))
         write(15,*) 'theta_initializing_b',noel,npt
          else if ((ei2(3).gt.0.0).and.(ei2(2).lt.0.0)) then
         write(15,*) 'theta_initializing:warning1'
           atheta=dacos(ei2(2))
          else if ((ei2(3).lt.0.0).and.(ei2(2).lt.0.0)) then
           atheta=-dacos(ei2(2))
         write(15,*) 'theta_initializing:warning2'
          else
           atheta=0.0
          endif
         else if (eigval(3).lt.eigval(2)) then
          if ((ei2(3).lt.0.0).and.(ei2(2).gt.0.0)) then
           atheta=(3.14159/2.0)-dacos(ei2(2))
         write(15,*) 'theta_initializing_c',noel,npt
          else if ((ei2(3).gt.0.0).and.(ei2(2).gt.0.0)) then
           atheta=-((3.14159/2.0)-dacos(ei2(2)))
         write(15,*) 'theta_initializing_d',noel,npt
          else if ((ei2(3).gt.0.0).and.(ei2(2).lt.0.0)) then
         write(15,*) 'theta_initializing:warning3'
           atheta=-dacos(ei2(2))
          else if ((ei2(3).lt.0.0).and.(ei2(2).lt.0.0)) then
           atheta=dacos(ei2(2))
         write(15,*) 'theta_initializing:warning4'
          else
           atheta=0.0
          endif
```

```
          else
         write(15,*)'voids are spherical:problems'
          endif
          atheta=0.0
          statev(9)=statev(9)+1.0
          write(15,*)'sending back'
          call flush(15)
          go to 99
         else
          atheta=0.0
          statev(9)=statev(9)+1.0
         endif
        endif
       endif
c------ Using the solution for 'dlam' (delta_lambda) to update all variables:
4      call update(dlam,ddsdde_mine)
       if ((f.lt.0.0).or.(wi1.lt.0.0).or.(wi2.lt.0.0)) then
        pnewdt=0.75
      write(15,*)'This should never happen: problem should have been
     &  taken care of before updating',dlam,f,wi1,wi2
      call flush(15)
        return
       endif
c------ stress and ddsdde are relative to the GLOBAL axes.
       s(1,1)=sc(1)
       s(2,2)=sc(2)
       s(3,3)=sc(3)
       s(1,2)=sc(4)/dsqrt(2.D0)
       s(2,3)=sc(5)/dsqrt(2.D0)
       s(3,1)=sc(6)/dsqrt(2.D0)
       s(2,1)=s(1,2)
       s(3,2)=s(2,3)
       s(1,3)=s(3,1)
       sc(1)=s(1,1)
       sc(2)=s(2,2)
       sc(3)=s(3,3)
       sc(4)=s(1,2)*dsqrt(2.D0)
       sc(5)=s(2,3)*dsqrt(2.D0)
       sc(6)=s(3,1)*dsqrt(2.D0)
       if (ntens.eq.4) then
        stress(1)=sc(2)
        stress(2)=sc(3)
        stress(3)=sc(1)
        stress(4)=sc(5)/dsqrt(2.D0)
       else
        stress(1)=sc(1)
        stress(2)=sc(2)
        stress(3)=sc(3)
        stress(4)=sc(4)/dsqrt(2.D0)
        stress(5)=sc(6)/dsqrt(2.D0)
        stress(6)=sc(5)/dsqrt(2.D0)
       endif
       statev(1)=f
       if ((statev(3).gt.1.0).and.(tottheta.lt.0.0)) then
        wi2=1.0/wi2
        wi1=wi2*wi1
        tottheta=0.5*3.14159+tottheta
       endif
       statev(2)=wi1
       statev(3)=wi2
       statev(4)=atheta
       statev(5)=aphi
       statev(6)=apsi
       statev(7)=eps_plastic
       statev(8)=sigy1
       statev(10)=hrd
       statev(11)=wi1/wi2
       statev(12)=rotj(1,1)
       statev(13)=rotj(1,2)
       statev(14)=rotj(1,3)
       statev(15)=rotj(2,1)
       statev(16)=rotj(2,2)
       statev(17)=rotj(2,3)
       statev(18)=rotj(3,1)
       statev(19)=rotj(3,2)
       statev(20)=rotj(3,3)
       tottheta=dasin(rot(3,2))
       statev(21)=tottheta*180.0/3.14159
       statev(22)=totphi*180.0/3.14159
       statev(23)=totpsi*180.0/3.14159
       statev(24)=1.0/wi2
       statev(25)=(1.0/wi2)*wi1
       statev(26)=1.000/wi1
c------ converting ddsdde for the case of 2D analyses:
```

```
         if (ntens.eq.4) then
        ddsdde(1,1)=ddsdde_mine(2,2)
        ddsdde(1,2)=ddsdde_mine(2,3)
        ddsdde(1,3)=ddsdde_mine(2,1)
        ddsdde(1,4)=ddsdde_mine(2,6)
        ddsdde(2,1)=ddsdde_mine(3,2)
        ddsdde(2,2)=ddsdde_mine(3,3)
        ddsdde(2,3)=ddsdde_mine(3,1)
        ddsdde(2,4)=ddsdde_mine(3,6)
        ddsdde(3,1)=ddsdde_mine(1,2)
        ddsdde(3,2)=ddsdde_mine(1,3)
        ddsdde(3,3)=ddsdde_mine(1,1)
        ddsdde(3,4)=ddsdde_mine(1,6)
        ddsdde(4,1)=ddsdde_mine(6,2)
        ddsdde(4,2)=ddsdde_mine(6,3)
        ddsdde(4,3)=ddsdde_mine(6,1)
        ddsdde(4,4)=ddsdde_mine(6,6)
         else
          do i=1,6
           do j=1,6
            ddsdde(i,j)=ddsdde_mine(i,j)
           enddo
          enddo
         endif
111     return
        end

c=================================================================================

        subroutine funcd(dlam,fun,dfun)
        real*8 fun,func,dlam,dfun
        logical debug,neg,check,yield
        common /mkdata2/yield,check,neg
        common /mdebug/debug
        external func

c
       fun=func(dlam)
       if (neg.eq..true.) then
        write(15,*)'Returning from FUNCD: Problem due to func evaluation'
        call flush(15)
        return
       endif
       dfun=(func(dlam+0.0001*dlam)-fun)/(0.0001*dlam)
       if (dfun.eq.0.0) then
        write(15,*)'You should never see this statement: Error trapping
     & incorrect in func'
        call flush(15)
        neg=.true.
       endif
       if (neg.eq..true.) then
        write(15,*)'Returning from FUNCD. Problem: In func during
     & slope calculation'
        call flush(15)
        return
       endif
       if(debug.eq..true.) then
       write(15,*)'dfun',dfun,fun,func(dlam+0.0001*dlam)
       endif
       return
       end

c=================================================================================
        subroutine update(dlam,ddsdde)
        real*8 s(3,3),MHS(6,6),Amat(6,6),Ce(6,6)
        real*8 sn(3,3),n(3,3),omega(3,3),sige(3,3),L
        real*8 strainc(6),sigec(6),ddsdde(6,6),sn1(3,3)
        real*8 sigma_n(6),sig_n(6),ddsdum(6,6)
        real*8 f,wi1,wi2,atheta,aphi,apsi,athetan,aphin,apsin
        real*8 fn,wi1n,wi2n,dlam,dumm(3,3,3,3)
        real*8 sc(6),nc(6),omegac(6),sigy1n
        real*8 mu1,k1,mu2,k2,sigy1,H,dumm1(3,3,3,3)
        real*8 dummy(6),a,b,c,ad,bd,cd,dummy1(3,3),dummy2(3,3)
        real*8 dummy5(6),dum1,nu1,nu2,sigy0
        real*8 deps(3,3),eps_plastic,dep_plas,dumd(6,6)
        real*8 det,k11,dphidf,phi1,phi2,ftest
        real*8 e1212,e2323,e1313,pi1212,pi2323,pi1313,bmat(6,6)
        real*8 mu1e,k1e,mu2e,ek2e,Ce_temp(6,6)
        real*8 ate,bte,dphidwi1,dphidwi2,wi1test,wi2test
        real*8 dum5,dum10,y,dinc(6),R(3,3),RT(3,3)
        real*8 cp,sp,ct,st,cs,ss,rot(3,3),rott(3,3)
        real*8 cpn,spn,ctn,stn,csn,ssn,rotn(3,3),rotnt(3,3)
        real*8 vec(3),hrd,rotj(3,3),rotjn(3,3),rota(3,3)
        real*8 test(6,6),loadp(6),loading
```

```
      real*8 zero(6),totstrain(6)
      integer i,j,ninc,indx(6),inpt
      logical path,yield,check,neg,spath,yc,debug
      logical evolf,evolwi,evolti
      common /controls/evolf,evolwi,evolti
      common /mkmodulus/mu1,k1,mu2,k2,sigy0,k11
      common /mkelas/mu1e,k1e,mu2e,ek2e
      common /mkdata1/f,wi1,wi2,sc,strainc,eps_plastic,sigy1,hrd
      common /mkorient/cp,sp,ct,st,cs,ss,rot,rott,atheta,aphi,apsi
      common /mkdata2/yield,check,neg
      common /mkrot/R,RT,rotj
      common /paths/spath
      common /mdebug/debug
c
      path=.true.
      neg=.false.
c
     if(debug.eq..true.) then
     write(15,*)'update1'
     call flush(15)
     endif
c    write(16,*)'strainc',strainc
     nu1=0.5*(3.0*k1-2.0*mu1)/(3.0*k1+mu1)
     nu2=0.5*(3.0*k2-2.0*mu2)/(3.0*k2+mu2)
      c=1.000
      a=c/wi1
      b=c/wi2
      ad=a
      bd=b
      cd=c
      if (yield.eq..false.) then
c----- The elastic predictor is the correct stress
c      The state variables are unchanged by elastic deformations
c      and so there is no need to update them.
       path=.false.
       call Meffective(a,b,c,ad,bd,cd,f,mu1e,k1e,mu2e,ek2e,Ce,path)
       path=.true.
       call tenmatprod(Ce,strainc,sigec)
       sige(1,1)=sc(1)+sigec(1)
       sige(2,2)=sc(2)+sigec(2)
       sige(3,3)=sc(3)+sigec(3)
       sige(1,2)=(sc(4)+sigec(4))/dsqrt(2.D0)
       sige(2,3)=(sc(5)+sigec(5))/dsqrt(2.D0)
       sige(3,1)=(sc(6)+sigec(6))/dsqrt(2.D0)
       sige(2,1)=sige(1,2)
       sige(3,2)=sige(2,3)
       sige(1,3)=sige(3,1)
c----- update stress (state variables are unchanged!)
c----- Express stress in GLOBAL coordinate frame.
       call matprod(rot,sige,dummy1)
       call matprod(dummy1,rott,sige)
       sc(1)=sige(1,1)
       sc(2)=sige(2,2)
       sc(3)=sige(3,3)
       sc(4)=sige(1,2)*dsqrt(2.D0)
       sc(5)=sige(2,3)*dsqrt(2.D0)
       sc(6)=sige(3,1)*dsqrt(2.D0)
c----- calculate ddsdde
       do 431 i=1,6
        do 441 j=1,6
         dumd(i,j)=Ce(i,j)
441     continue
431    continue
c------ Express ddsdde in GLOBAL coordinate frame
       call mat2tensor(dumd,dumm)
       call rot4order(dumm,rot,dumm1)
       call ten2matrix(dumm1,Ce_temp)
c------ Expressing ddsdde in the notation of ABAQUS
       do 432 i=1,3
        dumd(i,4)=Ce_temp(i,4)/dsqrt(2.D0)
        dumd(i,5)=Ce_temp(i,5)/dsqrt(2.D0)
        dumd(i,6)=Ce_temp(i,6)/dsqrt(2.D0)
432    continue
       do 433 i=4,6
        dumd(i,1)=Ce_temp(i,1)/dsqrt(2.D0)
        dumd(i,2)=Ce_temp(i,2)/dsqrt(2.D0)
        dumd(i,3)=Ce_temp(i,3)/dsqrt(2.D0)
433    continue
       do 434 i=4,6
        do 435 j=4,6
        dumd(i,j)=Ce_temp(i,j)/2.0
435     continue
434    continue
       do 436 i=1,4
```

```
          do 437 j=1,4
           ddsdde(i,j)=dumd(i,j)
437       continue
436      continue
         do 438 i=1,4
          ddsdde(i,5)=dumd(i,6)
          ddsdde(i,6)=dumd(i,5)
438      continue
         do 439 i=1,4
          ddsdde(5,i)=dumd(6,i)
          ddsdde(6,i)=dumd(5,i)
439      continue
          ddsdde(5,5)=dumd(6,6)
          ddsdde(5,6)=dumd(6,5)
          ddsdde(6,5)=dumd(5,6)
          ddsdde(6,6)=dumd(5,5)
         call ludcmp(Ce_temp,6,6,indx,det)
         do 544 j=1,6
          det=det*Ce_temp(j,j)
544      continue
         if (det.lt.0.0) then
c         write(15,*)'ddsdde_elastic',a,b,c
c         write(15,*)ddsdde
c         write(15,*)'det',det
         endif
         go to 999
        endif
c----- If the material has become plastic, the microstructural variables
c       need to be updated. Also the new stress has to be evaluated carefully
c       since the strains are no longer completely elastic. We must obtain
c       the plastic part of the strain (which is done by obtaining 'dlam'
c       i.e., delta_lambda) and then that is used to update all relevant
c       variables.
c------ calculate Meffective
        call Meffective(a,b,c,ad,bd,cd,f,mu1,k11,mu2,k2,MHS,path)
        do 100 i=1,6
         do 110 j=1,6
          MHS(i,j)=3.0*mu1*MHS(i,j)
110      continue
100     continue
c---------------------------------------------------------------------
c------ estimate stress
c------ STEP 1: Estimate the elastic predictor
        path=.false.
        call Meffective(a,b,c,ad,bd,cd,f,mu1e,k1e,mu2e,ek2e,Ce,path)
        path=.true.
        call tenmatprod(Ce,strainc,sigec)
       do 201, i=1,6
        loadp(i)=sigec(i)
201      continue
         sige(1,1)=sc(1)+sigec(1)
         sige(2,2)=sc(2)+sigec(2)
         sige(3,3)=sc(3)+sigec(3)
         sige(1,2)=(sc(4)+sigec(4))/dsqrt(2.D0)
         sige(2,3)=(sc(5)+sigec(5))/dsqrt(2.D0)
         sige(3,1)=(sc(6)+sigec(6))/dsqrt(2.D0)
         sige(2,1)=sige(1,2)
         sige(3,2)=sige(2,3)
         sige(1,3)=sige(3,1)
         sigec(1)=sige(1,1)
         sigec(2)=sige(2,2)
         sigec(3)=sige(3,3)
         sigec(4)=sige(1,2)*dsqrt(2.D0)
         sigec(5)=sige(2,3)*dsqrt(2.D0)
         sigec(6)=sige(3,1)*dsqrt(2.D0)
c------ STEP 1a: Determining the stress on the yield surface
         sigma_n(1)=sc(1)
         sigma_n(2)=sc(2)
         sigma_n(3)=sc(3)
         sigma_n(4)=sc(4)
         sigma_n(5)=sc(5)
         sigma_n(6)=sc(6)
         call yieldtest(MHS,sigma_n,sigy1,f,yc,y)
         if (yc.eq..false.) then
          call yieldtest(MHS,sigec,sigy1,f,yc,y)
          if (yc.eq..true.) then
c          this is the step where the behavior becomes plastic (the previous
c          step was elastic)
           check=.true.
         do 127 i=1,6
          zero(i)=0.0
          totstrain(i)=strainc(i)
127        continue
           ninc=100.0
```

```
          call stre(sigma_n,sigec,zero,totstrain,sig_n,Ce,ninc)
         else
       write(15,*)'NOT PLASTIC, shouldnt have come here'
       call flush(15)
          do 130 i=1,6
           sig_n(i)=sc(i)
130       continue
         endif
        else
          do 131 i=1,6
           sig_n(i)=sc(i)
131       continue
        endif
c------ calculate N(3,3)
       call tenmatprod(MHS,sig_n,nc)
        do 200 i=1,6
         nc(i)=nc(i)/(1.0-f)
         nc(i)=2.0*nc(i)/sigy1
         dummy(i)=nc(i)
         dummy5(i)=nc(i)
200     continue
c       write(16,*)'mhs',mhs(1,1),mhs(1,2),mhs(1,3),mhs(1,4)
c    &             ,mhs(1,5),mhs(1,6)
c       write(16,*)'mhs2',mhs(2,1),mhs(2,2),mhs(2,3),mhs(2,4)
c    &             ,mhs(2,5),mhs(2,6)
c       write(16,*)'sign',sig_n
c       write(16,*)'nc',nc
c       write(15,*)'exit stress'
c       write(15,*)sig_n
        n(1,1)=nc(1)
        n(2,2)=nc(2)
        n(3,3)=nc(3)
        n(1,2)=nc(4)/dsqrt(2.D0)
        n(2,1)=n(1,2)
        n(2,3)=nc(5)/dsqrt(2.D0)
        n(3,2)=n(2,3)
        n(3,1)=nc(6)/dsqrt(2.D0)
        n(1,3)=n(3,1)
c------ Test for loading/unloading
        call rowcolumnprod(loadp,dummy5,loading)
        do 203 i=1,6
         dummy5(i)=nc(i)
203     continue
        if (loading.le.0.0) then
       write(15,*)'loading',loading
        endif
c------ STEP 2: Remaining part of the stress
c------ STEP 2a: Calculating 'omega'
       spath=.false.
       call stensor(a,b,c,nu1,k1,mu1,dumd,pi1212,pi1313,pi2323)
       if (debug.eq..true.) then
       write(15,*)'pi1212',pi1212,pi2323,pi1313
       call flush(15)
       endif
       e1212=pi1212-f*pi1212
       e2323=pi2323-f*pi2323
       e1313=pi1313-f*pi1313
       spath=.true.
       call Btensor(a,b,c,ad,bd,cd,f,mu1,k1,mu2,k2,Bmat)
       do 41 i=1,3
        do 42 j=1,6
         Bmat(i,j)=0.0
42      continue
41     continue
       Bmat(4,1)=2.0*e1212*Bmat(4,1)
       Bmat(4,2)=2.0*e1212*Bmat(4,2)
       Bmat(4,3)=2.0*e1212*Bmat(4,3)
       Bmat(4,4)=2.0*e1212*Bmat(4,4)
       Bmat(4,5)=2.0*e1212*Bmat(4,5)
       Bmat(4,6)=2.0*e1212*Bmat(4,6)
       Bmat(5,1)=2.0*e2323*Bmat(5,1)
       Bmat(5,2)=2.0*e2323*Bmat(5,2)
       Bmat(5,3)=2.0*e2323*Bmat(5,3)
       Bmat(5,4)=2.0*e2323*Bmat(5,4)
       Bmat(5,5)=2.0*e2323*Bmat(5,5)
       Bmat(5,6)=2.0*e2323*Bmat(5,6)
       Bmat(6,1)=2.0*e1313*Bmat(6,1)
       Bmat(6,2)=2.0*e1313*Bmat(6,2)
       Bmat(6,3)=2.0*e1313*Bmat(6,3)
       Bmat(6,4)=2.0*e1313*Bmat(6,4)
       Bmat(6,5)=2.0*e1313*Bmat(6,5)
       Bmat(6,6)=2.0*e1313*Bmat(6,6)
       call Atensor(a,b,c,ad,bd,cd,f,mu1,k1,mu2,k2,Amat)
      do 3 i=1,3
```

```
        do 4 j=1,3
         omega(i,j)=0.0
4       continue
3      continue
       do 1 i=1,3
        omega(1,2)=-Bmat(4,i)*nc(i)+omega(1,2)
        omega(2,3)=-Bmat(5,i)*nc(i)+omega(2,3)
        omega(1,3)=-Bmat(6,i)*nc(i)+omega(1,3)
1      continue
       do 2 i=4,6
        omega(1,2)=-Bmat(4,i)*nc(i)+omega(1,2)
        omega(2,3)=-Bmat(5,i)*nc(i)+omega(2,3)
        omega(1,3)=-Bmat(6,i)*nc(i)+omega(1,3)
2      continue
       omega(2,1)=-omega(1,2)
       omega(3,2)=-omega(2,3)
       omega(3,1)=-omega(1,3)
       do 7 i=1,3
        do 8 j=1,3
         omega(i,j)=omega(i,j)/dsqrt(2.0d0)
8       continue
7      continue
       call tenmatprod(Amat,nc,dinc)
        c=1.000
        a=c/wi1
        b=c/wi2
        if (dabs(a-b).gt.0.01) then
         omega(1,2)=omega(1,2)-(a*a+b*b)*dinc(4)/(dsqrt(2.0d0)*(a*a-b*b))
        endif
        if (dabs(a-c).gt.0.01) then
         omega(1,3)=omega(1,3)-(a*a+c*c)*dinc(6)/(dsqrt(2.0d0)*(a*a-c*c))
        endif
        if (dabs(c-b).gt.0.01) then
         omega(2,3)=omega(2,3)-(b*b+c*c)*dinc(5)/(dsqrt(2.0d0)*(b*b-c*c))
        endif
        omega(2,1)=-omega(1,2)
        omega(3,2)=-omega(2,3)
        omega(3,1)=-omega(1,3)
        omegac(1)=omega(1,1)
        omegac(2)=omega(2,2)
        omegac(3)=omega(3,3)
        omegac(4)=omega(1,2)*dsqrt(2.D0)
        omegac(5)=omega(2,3)*dsqrt(2.D0)
        omegac(6)=omega(1,3)*dsqrt(2.D0)
c--------------------
        s(1,1)=sc(1)
        s(2,2)=sc(2)
        s(3,3)=sc(3)
        s(1,2)=sc(4)/dsqrt(2.D0)
        s(2,3)=sc(5)/dsqrt(2.D0)
        s(3,1)=sc(6)/dsqrt(2.D0)
        s(2,1)=s(1,2)
        s(3,2)=s(2,3)
        s(1,3)=s(3,1)
        call matprod(s,omega,dummy1)
        call matprod(omega,s,dummy2)
        do 300 i=1,3
         do 310 j=1,3
           sn(i,j)=dummy1(i,j)-dummy2(i,j)
310      continue
300     continue
        call tenmatprod(Ce,nc,dummy)
         sn(1,1)=sn(1,1)-dummy(1)
         sn(2,2)=sn(2,2)-dummy(2)
         sn(3,3)=sn(3,3)-dummy(3)
         sn(1,2)=sn(1,2)-dummy(4)/dsqrt(2.D0)
         sn(2,3)=sn(2,3)-dummy(5)/dsqrt(2.D0)
         sn(3,1)=sn(3,1)-dummy(6)/dsqrt(2.D0)
         sn(2,1)=sn(1,2)
         sn(3,2)=sn(2,3)
         sn(1,3)=sn(3,1)
         do 320 i=1,3
          do 330 j=1,3
           dummy1(i,j)=sn(i,j)
330       continue
320      continue
c------ STEP 3: put them together
         sn1(1,1)=sige(1,1)+dlam*sn(1,1)
         sn1(2,2)=sige(2,2)+dlam*sn(2,2)
         sn1(3,3)=sige(3,3)+dlam*sn(3,3)
         sn1(1,2)=sige(1,2)+dlam*sn(1,2)
         sn1(2,3)=sige(2,3)+dlam*sn(2,3)
         sn1(3,1)=sige(3,1)+dlam*sn(3,1)
         sn1(2,1)=sn1(1,2)
```

```
          sn1(3,2)=sn1(2,3)
          sn1(1,3)=sn1(3,1)
c------ These components of sn1 are now NOT in the coordinate farme
c------ coincides with the orientation of the inclusions. This is due
c------ the algorithm used (see notes for more details). The stress components
c         are now relative to a frame which is obtained by rotating the
c         orientation of the particles by R. So obtain the components relative
c         to the orientation of the particles at the beginning of this
c         increment:
          call matprod(R,sn1,dummy1)
          call matprod(dummy1,RT,sn1)
c-----------------------------------------------------------------------------
c----- estimate the new state variables
        if (evolf.eq..true.) then
          fn=f+(n(1,1)+n(2,2)+n(3,3))*(1.0-f)*dlam
        else
          fn=f
        endif
        if (inpt.eq.1) then
        endif
        call Atensor(a,b,c,ad,bd,cd,f,mu1,k1,mu2,k2,Amat)
        dummy(1)=Amat(3,1)-Amat(1,1)
        dummy(2)=Amat(3,2)-Amat(1,2)
        dummy(3)=Amat(3,3)-Amat(1,3)
        dummy(4)=Amat(3,4)-Amat(1,4)
        dummy(5)=Amat(3,5)-Amat(1,5)
        dummy(6)=Amat(3,6)-Amat(1,6)
        call rowcolumnprod(dummy,dummy5,dum5)
c       write(16,*)'dummy1',dummy
c       write(16,*)'dummy5',dummy5
c       write(16,*),'A1',Amat(3,1)
c       write(16,*),'A1',Amat(3,2)
c       write(16,*),'A1',Amat(3,3)
c       write(16,*),'A1',Amat(3,4)
c       write(16,*),'A1',Amat(3,5)
c       write(16,*),'A1',Amat(3,6)
c       write(16,*),'A1',Amat(1,1)
c       write(16,*),'A1',Amat(1,2)
c       write(16,*),'A1',Amat(1,3)
c       write(16,*),'A1',Amat(1,4)
c       write(16,*),'A1',Amat(1,5)
c       write(16,*),'A1',Amat(1,6)
        do 400 i=1,6
          dummy5(i)=nc(i)
400     continue
        dummy(1)=Amat(3,1)-Amat(2,1)
        dummy(2)=Amat(3,2)-Amat(2,2)
        dummy(3)=Amat(3,3)-Amat(2,3)
        dummy(4)=Amat(3,4)-Amat(2,4)
        dummy(5)=Amat(3,5)-Amat(2,5)
        dummy(6)=Amat(3,6)-Amat(2,6)
        call rowcolumnprod(dummy,dummy5,dum10)
c       write(16,*)'dummy2',dummy
c       write(16,*)'dummy52',dummy5
c       write(16,*),'A2',Amat(3,1)
c       write(16,*),'A2',Amat(3,2)
c       write(16,*),'A2',Amat(3,3)
c       write(16,*),'A2',Amat(3,4)
c       write(16,*),'A2',Amat(3,5)
c       write(16,*),'A2',Amat(3,6)
c       write(16,*),'A2',Amat(2,1)
c       write(16,*),'A2',Amat(2,2)
c       write(16,*),'A2',Amat(2,3)
c       write(16,*),'A2',Amat(2,4)
c       write(16,*),'A2',Amat(2,5)
c       write(16,*),'A2',Amat(2,6)
       if (evolwi.eq..true.) then
        wi1n=wi1+dlam*wi1*dum5
        wi2n=wi2+dlam*wi2*dum10
c       write(15,*)'win',wi1n,wi2n
c       write(15,*)'wi1',wi1,wi2,dum5,dum10
       else
        wi1n=wi1
        wi2n=wi2
       endif
        if ((wi1n.le.0.0).or.(wi2n.le.0.0)) then
       write(15,*)'Aspects have become negative'
       call flush(15)
         neg=.true.
         return
        endif
c----- estimating new orientations
        if (evolti.eq..true.) then
         athetan=atheta-(omega(2,3)*cp+omega(1,3)*sp)*dlam
```

```
      if (dabs(wi2n-1.0).lt.0.01) then
       athetan=0.0
        endif
        if (st.ne.0.0) then
         aphin=aphi-((omega(2,3)*ss/st)-omega(1,3)*cs/st)*dlam
         apsin=apsi+(-omega(1,2)+(ct/st)*(omega(2,3)*ss-omega(1,3)*cs))*
     &               dlam
        else
         apsin=0.0
         aphin=0.0
        endif
      call matprod(R,rotj,rotjn)
       else
        aphin=aphi
        apsin=apsi
        athetan=atheta
      do 27 i=1,3
       do 28 j=1,3
        rotjn(i,j)=rotj(i,j)
28       continue
27      continue
       endif
       ctn=dcos(athetan)
       stn=dsin(athetan)
       cpn=dcos(aphin)
       csn=dcos(apsin)
       spn=dsin(aphin)
       ssn=dsin(apsin)
       rota(1,1)=csn*cpn-ssn*ctn*spn
       rota(1,2)=-csn*spn-ssn*ctn*cpn
       rota(1,3)=ssn*stn
       rota(2,1)=ssn*cpn+csn*ctn*spn
       rota(2,2)=-ssn*spn+csn*ctn*cpn
       rota(2,3)=-csn*stn
       rota(3,1)=stn*spn
       rota(3,2)=stn*cpn
       rota(3,3)=ctn
       call matprod(rota,rotjn,rotn)
        do 77 i=1,3
         do 88 j=1,3
          rotnt(i,j)=rotn(j,i)
88        continue
77       continue
c-----------------------------------------------------------------------
c----- update stress and state variables:
c      The stress must now be expressed w.r.t the new orientation of the
c      particles so that ddsdde can be calculated.
c      First we take it back w.r.t the GLOBAL axes:
       call matprod(rot,sn1,dummy)
       call matprod(dummy,rott,sn1)
c      and then express it in w.r.t the new orientation of the particles.
       call matprod(rotnt,sn1,dummy)
       call matprod(dummy,rotn,sn1)
       sc(1)=sn1(1,1)
       sc(2)=sn1(2,2)
       sc(3)=sn1(3,3)
       sc(4)=sn1(1,2)*dsqrt(2.D0)
       sc(5)=sn1(2,3)*dsqrt(2.D0)
       sc(6)=sn1(3,1)*dsqrt(2.D0)
       f=fn
       wi1=wi1n
       wi2=wi2n
       c=1.000
       a=c/wi1
       b=c/wi2
       ad=a
       bd=b
       cd=c
       do 37 i=1,3
      do 38 j=1,3
       rotj(i,j)=rotjn(i,j)
38      continue
37     continue
       atheta=athetan
       aphi=aphin
       apsi=apsin
        do 707 i=1,3
         do 808 j=1,3
          rot(i,j)=rotn(i,j)
          rott(i,j)=rotnt(i,j)
808        continue
707      continue
c----- updating sig_yield
       do 501 i=1,3
```

```
         do 502 j=1,3
          deps(i,j)=dlam*n(i,j)
502      continue
501     continue
        dum1=(deps(1,1)+deps(2,2)+deps(3,3))/3.0
        do 503 i=1,3
         deps(i,i)=deps(i,i)-dum1
503     continue
        dep_plas=deps(1,1)**2+deps(2,2)**2+deps(3,3)**2
        dep_plas=dep_plas+2.0*(deps(1,2)**2+deps(2,3)**2+deps(3,1)**2)
        dep_plas=(2.0/3.0)*dep_plas
        eps_plastic=eps_plastic+dsqrt(dep_plas)
        sigy1n=sigy0*((1.0+eps_plastic)**(1.0/3.0))
C       sigy1=sigy1n
           sigy1n=sigy1
c-----------------------------------------------------------------------
        call Meffective(a,b,c,ad,bd,cd,f,mu1,k11,mu2,k2,MHS,path)
        do 391 i=1,6
         do 392 j=1,6
          MHS(i,j)=3.0*mu1*MHS(i,j)
392      continue
391     continue
        call yieldtest(MHS,sc,sigy1,f,yc,y)
c
c----- CALCULATE ddsdde
c----- calculate N
        do 399 i=1,6
         dummy5(i)=sc(i)
399     continue
        call tenmatprod(MHS,dummy5,nc)
         do 401 i=1,6
          nc(i)=nc(i)/(1.0-f)
          nc(i)=2.0*nc(i)/sigy1
401      continue
        path=.false.
        call Meffective(a,b,c,ad,bd,cd,f,mu1e,k1e,mu2e,ek2e,Ce,path)
        path=.true.
        do 470 i=1,6
         do 471 j=1,6
          test(i,j)=Ce(i,j)
471      continue
470     continue
        call ludcmp(test,6,6,indx,det)
        do 443 j=1,6
         det=det*test(j,j)
443     continue
        if (det.lt.0.0) then
       write(15,*)'Ce is wrong during plastic step'
       call flush(15)
        endif
        do 410 i=1,6
         dummy5(i)=nc(i)
410     continue
        call tenmatprod(Ce,dummy5,dummy)
        do 420 i=1,6
         dummy5(i)=nc(i)
420     continue
        call rowcolumnprod(dummy,dummy5,L)
c----- calculating H
c----- step A -- calculating dphi/df
         H=0.0
         path=.true.
         if (evolf.eq..true.) then
         call Meffective(a,b,c,ad,bd,cd,f,mu1,k11,mu2,k2,MHS,path)
         do 1001 i=1,6
          do 1002 j=1,6
           MHS(i,j)=3.0*mu1*MHS(i,j)
1002       continue
1001      continue
         call yieldtest(MHS,sc,sigy1,f,yc,phi1)
         ftest=f+0.001*f
         call Meffective(a,b,c,ad,bd,cd,ftest,mu1,k11,mu2,k2,MHS,path)
         do 1003 i=1,6
          do 1004 j=1,6
           MHS(i,j)=3.0*mu1*MHS(i,j)
1004       continue
1003      continue
         call yieldtest(MHS,sc,sigy1,ftest,yc,phi2)
         dphidf=(phi2-phi1)/(ftest-f)
         H=0.0
         H=-dphidf*(1.0-f)*(nc(1)+nc(2)+nc(3))
        endif
c----- step B -- calculating dphi/dwi1
         if (evolwi.eq..true.) then
```

```
       path=.true.
       call Meffective(a,b,c,ad,bd,cd,f,mu1,k11,mu2,k2,MHS,path)
       do 2001 i=1,6
        do 2002 j=1,6
         MHS(i,j)=3.0*mu1*MHS(i,j)
2002     continue
2001    continue
       call yieldtest(MHS,sc,sigy1,f,yc,phi1)
       wi1test=wi1+0.001*wi1
       ate=c/wi1test
       call Meffective(ate,b,c,ate,b,c,f,mu1,k11,mu2,k2,
     &                                       MHS,path)
       do 2003 i=1,6
        do 2004 j=1,6
         MHS(i,j)=3.0*mu1*MHS(i,j)
2004     continue
2003    continue
       call yieldtest(MHS,sc,sigy1,f,yc,phi2)
       dphidwi1=(phi2-phi1)/(wi1test-wi1)
       H=H-dphidwi1*wi1*dum5
      endif
c----- step C -- calculating dphi/dwi2
      if (evolwi.eq..true.) then
       path=.true.
       call Meffective(a,b,c,ad,bd,cd,f,mu1,k11,mu2,k2,MHS,path)
       do 3001 i=1,6
        do 3002 j=1,6
         MHS(i,j)=3.0*mu1*MHS(i,j)
3002     continue
3001    continue
       call yieldtest(MHS,sc,sigy1,f,yc,phi1)
       wi2test=wi2+0.001*wi2
       bte=c/wi2test
       call Meffective(a,bte,c,a,bte,c,f,mu1,k11,mu2,k2,
     &                          MHS,path)
       do 3003 i=1,6
        do 3004 j=1,6
         MHS(i,j)=3.0*mu1*MHS(i,j)
3004     continue
3003    continue
       call yieldtest(MHS,sc,sigy1,f,yc,phi2)
       dphidwi2=(phi2-phi1)/(wi2test-wi2)
       H=H-dphidwi2*wi2*dum10
      endif
     if(debug.eq..true.) then
     write(15,*)'update8'
     call flush(15)
     endif
     hrd=H
c---------------------------------------------
      L=L+H
c     write(15,*)'L',L,'H',H
      call flush(15)
      if (L.lt.0.0) then
       write(15,*)'L is negative'
       call flush(15)
      endif
c-----------------------------------
      s(1,1)=sc(1)
      s(2,2)=sc(2)
      s(3,3)=sc(3)
      s(1,2)=sc(4)/dsqrt(2.D0)
      s(2,3)=sc(5)/dsqrt(2.D0)
      s(3,1)=sc(6)/dsqrt(2.D0)
      s(2,1)=s(1,2)
      s(3,2)=s(2,3)
      s(1,3)=s(3,1)
      do 450 i=1,6
       dummy5(i)=nc(i)
450   continue
      call tenmatprod(Ce,dummy5,dummy)
      call columnrowprod(dummy,dummy,ddsdde)
      do 430 i=1,6
       do 440 j=1,6
        ddsdde(i,j)=Ce(i,j)-ddsdde(i,j)/L
440    continue
430   continue
c-----
      do 111 i=1,6
       do 112 j=1,6
        test(i,j)=ddsdde(i,j)
112    continue
111   continue
      call ludcmp(test,6,6,indx,det)
```

```
      do 113 j=1,6
       det=det*test(j,j)
113   continue
      if (det.lt.0.0) then
c      write(15,*)'det1',det
       call flush(15)
      endif
      call matprod(s,omega,dummy1)
      call matprod(omega,s,dummy2)
      do 421 i=1,3
       do 422 j=1,3
        dummy1(i,j)=dummy1(i,j)-dummy2(i,j)
ccc         dummy1(i,j)=0.0
422    continue
421   continue
      dummy5(1)=dummy1(1,1)
      dummy5(2)=dummy1(2,2)
      dummy5(3)=dummy1(3,3)
      dummy5(4)=dummy1(1,2)*dsqrt(2.D0)
      dummy5(5)=dummy1(2,3)*dsqrt(2.D0)
      dummy5(6)=dummy1(3,1)*dsqrt(2.D0)
      call columnrowprod(dummy5,dummy,ddsdum)
      if (evolf.eq..true.) then
      do 4431 i=1,6
       do 4432 j=1,6
        ddsdde(i,j)=ddsdde(i,j)+ddsdum(i,j)/L
4432    continue
4431   continue
      endif
c-----Express components of ddsdde w.r.t fixed GLOBAL axes.
     call mat2tensor(ddsdde,dumm)
     call rot4order(dumm,rotn,dumm1)
     call ten2matrix(dumm1,ddsdde)
      do 570 i=1,6
       do 571 j=1,6
        dumd(i,j)=ddsdde(i,j)
571    continue
570   continue
      call ludcmp(dumd,6,6,indx,det)
      do 543 j=1,6
       det=det*dumd(j,j)
543   continue
      if (det.lt.0.0) then
c      write(15,*)'abc',a,b,c
c      write(15,*)ddsdde
c      write(15,*)'
c      write(15,*)'Ce'
c      write(15,*)Ce
c      write(15,*)'det',det
      endif
c---- Express stress components w.r.t GLOBAL axes.
     if (evolti.eq..true.) then
      call matprod(rotn,s,dummy1)
      call matprod(dummy1,rotnt,s)
     endif
       sc(1)=s(1,1)
       sc(2)=s(2,2)
       sc(3)=s(3,3)
       sc(4)=s(1,2)*dsqrt(2.D0)
       sc(5)=s(2,3)*dsqrt(2.D0)
       sc(6)=s(3,1)*dsqrt(2.D0)
c----  If the constitutive equation is written in terms of the
c      Jaumann rate of the Cauchy stress, then ddsdde will be
c      unsymmetric. The additional quantity that appears in ddsdde
c      is added below.
c      write(15,*)'dds11',ddsdde(1,1),s(1,1)
c      write(15,*)'dds22',ddsdde(2,2),s(2,2)
c      write(15,*)'dds33',ddsdde(3,3),s(3,3)
c      write(15,*)'dds44',ddsdde(4,1),sc(4)
c      write(15,*)'dds55',ddsdde(5,1),sc(5)
c      write(15,*)'dds66',ddsdde(6,1),sc(6)
c      call flush(15)
c      ddsdde(1,1)=ddsdde(1,1)+s(1,1)
c      ddsdde(1,2)=ddsdde(1,2)+s(1,1)
c      ddsdde(1,3)=ddsdde(1,3)+s(1,1)
c      ddsdde(2,1)=ddsdde(2,1)+s(2,2)
c      ddsdde(2,2)=ddsdde(2,2)+s(2,2)
c      ddsdde(2,3)=ddsdde(2,3)+s(2,2)
c      ddsdde(3,1)=ddsdde(3,1)+s(3,3)
c      ddsdde(3,2)=ddsdde(3,2)+s(3,3)
c      ddsdde(3,3)=ddsdde(3,3)+s(3,3)
c      ddsdde(4,1)=ddsdde(4,1)+sc(4)
c      ddsdde(4,2)=ddsdde(4,2)+sc(4)
c      ddsdde(4,3)=ddsdde(4,3)+sc(4)
```

```
c      ddsdde(5,1)=ddsdde(5,1)+sc(5)
c      ddsdde(5,2)=ddsdde(5,2)+sc(5)
c      ddsdde(5,3)=ddsdde(5,3)+sc(5)
c      ddsdde(6,1)=ddsdde(6,1)+sc(6)
c      ddsdde(6,2)=ddsdde(6,2)+sc(6)
c      ddsdde(6,3)=ddsdde(6,3)+sc(6)
C---- Interchange components of ddsdde here (since our notation for
c     stress and strain are different from what they use).
       do 477 i=1,3
        do 478 j=4,6
         ddsdde(i,j)=ddsdde(i,j)/dsqrt(2.D0)
478     continue
477    continue
       do 473 i=4,6
        do 474 j=1,3
         ddsdde(i,j)=ddsdde(i,j)/dsqrt(2.D0)
474     continue
473    continue
       do 475 i=4,6
        do 476 j=4,6
         ddsdde(i,j)=ddsdde(i,j)/(2.D0)
476     continue
475    continue
       do 530 i=1,6
        do 531 j=1,6
         dumd(i,j)=ddsdde(i,j)
531     continue
530    continue
       do 538 i=1,4
        ddsdde(i,5)=dumd(i,6)
        ddsdde(i,6)=dumd(i,5)
538    continue
       do 539 i=1,4
        ddsdde(5,i)=dumd(6,i)
        ddsdde(6,i)=dumd(5,i)
539    continue
        ddsdde(5,5)=dumd(6,6)
        ddsdde(5,6)=dumd(6,5)
        ddsdde(6,5)=dumd(5,6)
        ddsdde(6,6)=dumd(5,5)
c------ The above is the final ddsdde
       do 541 i=1,6
        do 542 j=1,6
         dumd(i,j)=ddsdde(i,j)
542     continue
541    continue
      if(debug.eq..true.) then
      write(15,*)'update10'
      call flush(15)
      endif
999   return
      end
c=======================================================================

       subroutine stre(ss,sse,p,q,estimate,Ce,ninc)
       real*8 MHS(6,6)
       real*8 p(6),q(6),r(6),eps_plastic,Ce(6,6)
       real*8 mu1,k1,mu2,k2,sigy1,f,wi1,wi2,sc(6),strainc(6)
       real*8 a,b,c,ad,bd,cd,sigy0,k11,y,hrd,ss(6),sse(6),estimate(6)
       integer ninc,i,j,ic
       logical path,check,yield,neg,yc,debug
       logical evolf,evolwi,evolti
       common /controls/evolf,evolwi,evolti
       common /mkmodulus/mu1,k1,mu2,k2,sigy0,k11
       common /mkdata1/f,wi1,wi2,sc,strainc,eps_plastic,sigy1,hrd
       common /mkdata2/yield,check,neg
       common /mdebug/debug
       path=.true.
       c=1.000
       a=c/wi1
       b=c/wi2
       ad=a
       bd=b
       cd=c
       call Meffective(a,b,c,ad,bd,cd,f,mu1,k11,mu2,k2,MHS,path)
       do 1 i=1,6
        do 2 j=1,6
         MHS(i,j)=3.0*mu1*MHS(i,j)
2       continue
1      continue
        if (check.eq..true.) then
         call yieldtest(MHS,ss,sigy1,f,yc,y)
         if (yc.eq..true.) then
          write(15,*)'WARNING1 IN STRE',y
```

```
          stop
         endif
         call yieldtest(MHS,sse,sigy1,f,yc,y)
         if (yc.eq..false.) then
          write(15,*)'WARNING2 IN STRE'
          stop
         endif
        endif
       do 100 ic=1,ninc
        do 10 i=1,6
          r(i)=0.5*(p(i)+q(i))
10      continue
        call tenmatprod(Ce,r,estimate)
      do 11 i=1,6
       estimate(i)=estimate(i)+ss(i)
11      continue
        call yieldtest(MHS,estimate,sigy1,f,yc,y)
cwrite(15,*)'est in stre',y
        if ((yc.eq..true.).and.(y.lt.1.0d-10))then
         return
        else
         if (yc.eq..false.) then
          do 20 i=1,6
           p(i)=r(i)
20        continue
         else
          do 30 i=1,6
           q(i)=r(i)
30        continue
         endif
        endif
100    continue
       write(15,*)'problem in stre'
       stop
       return
       end
c================================================================================

c------ This is used to decompose the F tensor.
        subroutine decompose(F,R,U,C,D)
        real*8 F(3,3),R(3,3),FT(3,3),C(3,3)
        real*8 D(3),V(3,3),U(3,3)
        integer n,np,nrot,i,j
      logical debug
      common /mdebug/debug
        do 10 i=1,3
         do 20 j=1,3
          FT(i,j)=F(j,i)
20       continue
10      continue
        call matprod(FT,F,C)
        n=3
        np=3
        call jacobi(C,n,np,D,V,nrot)
        do 30 i=1,3
         do 40 j=1,3
          C(i,j)=0.0
        U(i,j)=0.0
40       continue
30      continue
        do 50 i=1,3
         do 60 j=1,3
          FT(i,j)=V(i,1)*V(j,1)
          U(i,j)=FT(i,j)*dlog(dsqrt(d(1)))
          C(i,j)=FT(i,j)/dsqrt(d(1))
60       continue
50      continue
        do 70 i=1,3
         do 80 j=1,3
          FT(i,j)=V(i,2)*V(j,2)
          U(i,j)=(FT(i,j)*dlog(dsqrt(d(2))))+U(i,j)
          C(i,j)=(FT(i,j)/dsqrt(d(2)))+C(i,j)
80       continue
70      continue
        do 90 i=1,3
         do 100 j=1,3
          FT(i,j)=V(i,3)*V(j,3)
          U(i,j)=(FT(i,j)*dlog(dsqrt(d(3))))+U(i,j)
          C(i,j)=(FT(i,j)/dsqrt(d(3)))+C(i,j)
100       continue
90      continue
        call matprod(F,C,R)
      call matprod(R,V,C)
      C(1,1)=C(1,1)/dsqrt(C(1,1)**2+C(2,1)**2+C(3,1)**2)
```

```
      C(2,1)=C(2,1)/dsqrt(C(1,1)**2+C(2,1)**2+C(3,1)**2)
      C(3,1)=C(3,1)/dsqrt(C(1,1)**2+C(2,1)**2+C(3,1)**2)
      C(1,2)=C(1,2)/dsqrt(C(1,2)**2+C(2,2)**2+C(3,2)**2)
      C(2,2)=C(2,2)/dsqrt(C(1,2)**2+C(2,2)**2+C(3,2)**2)
      C(3,2)=C(3,2)/dsqrt(C(1,2)**2+C(2,2)**2+C(3,2)**2)
      C(1,3)=C(1,3)/dsqrt(C(1,3)**2+C(2,3)**2+C(3,3)**2)
      C(2,3)=C(2,3)/dsqrt(C(1,3)**2+C(2,3)**2+C(3,3)**2)
      C(3,3)=C(3,3)/dsqrt(C(1,3)**2+C(2,3)**2+C(3,3)**2)
        return
        end

c------ This is used to decompose the F tensor.
        subroutine decompose1(F,R,U)
        real*8 F(3,3),R(3,3),FT(3,3),C(3,3)
        real*8 D(3),V(3,3),U(3,3),Uinv(3,3)
        integer n,np,nrot,i,j
      logical debug
      common /mdebug/debug
        do 10 i=1,3
         do 20 j=1,3
          FT(i,j)=F(j,i)
20       continue
10      continue
        call matprod(FT,F,C)
        n=3
        np=3
        call jacobi(C,n,np,D,V,nrot)
        do 30 i=1,3
         do 40 j=1,3
          C(i,j)=0.0
40       continue
30      continue
        do 50 i=1,3
         do 60 j=1,3
          FT(i,j)=V(i,1)*V(j,1)
          U(i,j)=FT(i,j)*(dsqrt(d(1)))
          C(i,j)=FT(i,j)/dsqrt(d(1))
60       continue
50      continue
        do 70 i=1,3
         do 80 j=1,3
          FT(i,j)=V(i,2)*V(j,2)
          U(i,j)=(FT(i,j)*(dsqrt(d(2))))+U(i,j)
          C(i,j)=(FT(i,j)/dsqrt(d(2)))+C(i,j)
80       continue
70      continue
        do 90 i=1,3
         do 100 j=1,3
          FT(i,j)=V(i,3)*V(j,3)
          U(i,j)=(FT(i,j)*(dsqrt(d(3))))+U(i,j)
          C(i,j)=(FT(i,j)/dsqrt(d(3)))+C(i,j)
100       continue
90      continue
        do 110 i=1,3
         do 120 j=1,3
          V(i,j)=U(i,j)
120      continue
110     continue
        call inverse3x3(V,Uinv)
        call matprod(F,Uinv,R)
        return
        end

c------ This is a routine to multiply 2nd order tensors (3x3 matrices)
        subroutine matprod(p,q,r)
        real*8 p(3,3),q(3,3),r(3,3)
        integer i,j,k
        do 10 i=1,3
         do 11 j=1,3
          r(i,j)=0.0
11       continue
10      continue
        do 12 i=1,3
         do 13 j=1,3
          do 14 k=1,3
           r(i,j)=r(i,j)+p(i,k)*q(k,j)
14        continue
13       continue
12      continue
        return
        end
```

```
      SUBROUTINE JACOBI(A,N,NP,D,V,NROT)
      implicit double precision (a-h, o-z)
      integer nmax
      PARAMETER (NMAX=100)
      real*8 A(NP,NP),D(NP),V(NP,NP),B(NMAX),Z(NMAX)
      integer ip,iq,i,n,np,nrot,j
      DO 12 IP=1,N
        DO 11 IQ=1,N
          V(IP,IQ)=0.
11      CONTINUE
        V(IP,IP)=1.
12    CONTINUE
      DO 13 IP=1,N
        B(IP)=A(IP,IP)
        D(IP)=B(IP)
        Z(IP)=0.
13    CONTINUE
      NROT=0
      DO 24 I=1,50
        SM=0.
        DO 15 IP=1,N-1
          DO 14 IQ=IP+1,N
            SM=SM+DABS(A(IP,IQ))
14        CONTINUE
15      CONTINUE
        IF(SM.EQ.0.)RETURN
        IF(I.LT.4)THEN
          TRESH=0.2*SM/N**2
        ELSE
          TRESH=0.
        ENDIF
        DO 22 IP=1,N-1
          DO 21 IQ=IP+1,N
            G=100.*DABS(A(IP,IQ))
            IF((I.GT.4).AND.(DABS(D(IP))+G.EQ.DABS(D(IP)))
     *         .AND.(DABS(D(IQ))+G.EQ.DABS(D(IQ))))THEN
              A(IP,IQ)=0.
            ELSE IF(DABS(A(IP,IQ)).GT.TRESH)THEN
              H=D(IQ)-D(IP)
              IF(DABS(H)+G.EQ.DABS(H))THEN
                T=A(IP,IQ)/H
              ELSE
                THETA=0.5*H/A(IP,IQ)
                T=1./(DABS(THETA)+DSQRT(1.D0+THETA**2))
                IF(THETA.LT.0.)T=-T
              ENDIF
              C=1./DSQRT(1.D0+T**2)
              S=T*C
              TAU=S/(1.+C)
              H=T*A(IP,IQ)
              Z(IP)=Z(IP)-H
              Z(IQ)=Z(IQ)+H
              D(IP)=D(IP)-H
              D(IQ)=D(IQ)+H
              A(IP,IQ)=0.
              DO 16 J=1,IP-1
                G=A(J,IP)
                H=A(J,IQ)
                A(J,IP)=G-S*(H+G*TAU)
                A(J,IQ)=H+S*(G-H*TAU)
16            CONTINUE
              DO 17 J=IP+1,IQ-1
                G=A(IP,J)
                H=A(J,IQ)
                A(IP,J)=G-S*(H+G*TAU)
                A(J,IQ)=H+S*(G-H*TAU)
17            CONTINUE
              DO 18 J=IQ+1,N
                G=A(IP,J)
                H=A(IQ,J)
                A(IP,J)=G-S*(H+G*TAU)
                A(IQ,J)=H+S*(G-H*TAU)
18            CONTINUE
              DO 19 J=1,N
                G=V(J,IP)
                H=V(J,IQ)
                V(J,IP)=G-S*(H+G*TAU)
                V(J,IQ)=H+S*(G-H*TAU)
19            CONTINUE
              NROT=NROT+1
            ENDIF
21        CONTINUE
22      CONTINUE
        DO 23 IP=1,N
```

```
         B(IP)=B(IP)+Z(IP)
         D(IP)=B(IP)
         Z(IP)=0.
23     CONTINUE
24    CONTINUE
      PAUSE '50 iterations should never happen'
      RETURN
      END


C----This is to calculate the A tensor
      subroutine Atensor(a,b,c,ad,bd,cd,c2,mu1,k1,mu2,k2,Amat)
      real*8 Amat(6,6),L1(6,6),L2(6,6),Si(6,6)
      real*8 M1(6,6),mu1,mu2,k1,k2,nu1,nu2
      real*8 a,b,c,ad,bd,cd,c1,c2
      real*8 d(6,6),d1(6,6),d2(6,6),pi1212,pi2323,pi1313
      integer i,j
      logical debug,spath
      common /paths/spath
      common /mdebug/debug
      spath=.true.
      nu1=0.5*(3.0*k1-2.0*mu1)/(3.0*k1+mu1)
      nu2=0.5*(3.0*k2-2.0*mu2)/(3.0*k2+mu2)
      c1=1.0-c2
      if(debug.eq..true.) then
      write(15,*)'Aten1'
      call flush(15)
      endif
c     Initialize the modulus tensors
      do 10 i=1,6
       do 20 j=1,6
        L1(i,j)=0.0
        L2(i,j)=0.0
        d(i,j)=0.0
20     continue
10    continue
c     Obtain the (6x6) modulus tensors, L1 and L2
      L1(1,1)=k1+(4.0/3.0)*mu1
      L1(2,2)=k1+(4.0/3.0)*mu1
      L1(3,3)=k1+(4.0/3.0)*mu1
      L1(1,2)=k1-(2.0/3.0)*mu1
      L1(2,1)=k1-(2.0/3.0)*mu1
      L1(2,3)=k1-(2.0/3.0)*mu1
      L1(3,2)=k1-(2.0/3.0)*mu1
      L1(3,1)=k1-(2.0/3.0)*mu1
      L1(1,3)=k1-(2.0/3.0)*mu1
      L1(4,4)=2.0*mu1
      L1(5,5)=2.0*mu1
      L1(6,6)=2.0*mu1
      L2(1,1)=k2+(4.0/3.0)*mu2
      L2(2,2)=k2+(4.0/3.0)*mu2
      L2(3,3)=k2+(4.0/3.0)*mu2
      L2(1,2)=k2-(2.0/3.0)*mu2
      L2(2,1)=k2-(2.0/3.0)*mu2
      L2(2,3)=k2-(2.0/3.0)*mu2
      L2(3,2)=k2-(2.0/3.0)*mu2
      L2(3,1)=k2-(2.0/3.0)*mu2
      L2(1,3)=k2-(2.0/3.0)*mu2
      L2(4,4)=2.0*mu2
      L2(5,5)=2.0*mu2
      L2(6,6)=2.0*mu2
c     Begin calculating A tensor
      if(debug.eq..true.) then
      write(15,*)'Aten2'
      call flush(15)
      endif
      call inverse(L1,M1)
      call tenprod(M1,L2,d)
      do 30 i=1,6
       d(i,i)=d(i,i)-1.0
30    continue
      do 31 i=1,6
       do 32 j=1,6
        d1(i,j)=0.0
        d2(i,j)=0.0
32     continue
31    continue
c     calculate S tensors
      if(debug.eq..true.) then
      write(15,*)'Aten3'
      call flush(15)
      endif
      call stensor(a,b,c,nu1,k1,mu1,Si,pi1212,pi1313,pi2323)
      if(debug.eq..true.) then
```

```
      write(15,*)'Aten4'
      call flush(15)
      endif
      do 60 i=1,6
       do 70 j=1,6
        d1(i,j)=Si(i,j)-c2*Si(i,j)
70     continue
60    continue
      call tenprod(d1,d,d2)
      do 80 i=1,6
       d2(i,i)=d2(i,i)+1.0
80    continue
      if(debug.eq..true.) then
      write(15,*)'Aten5',d2
      call flush(15)
      endif
      call inverse(d2,Amat)
      if(debug.eq..true.) then
      write(15,*)'Aten6', Amat
      call flush(15)
      endif
      return
      end


      subroutine columnrowprod(a,b,c)
      real*8 a(6),b(6),c(6,6)
      integer i,j
      do 10 i=1,6
       do 20 j=1,6
        c(i,j)=a(i)*b(j)
20     continue
10    continue
      return
      end


C----This is to calculate the effective modulus of the composite.
      subroutine Meffective(a,b,c,ad,bd,cd,c2,mu1,k1,mu2,k2,MHS,path)
      real*8 LHS(6,6),L1(6,6),L2(6,6),Si(6,6),MHS(6,6)
      real*8 M1(6,6),mu1,mu2,k1,k2,nu1,nu2
      real*8 a,b,c,ad,bd,cd,c1,c2
      real*8 d(6,6),d1(6,6),pi1212,pi2323,pi1313
      integer i,j
      logical path,spath,debug
      common /paths/spath
      common /mdebug/debug
      spath=.true.
      if(debug.eq..true.) then
        write(15,*)'Meff1'
        call flush(15)
      endif
      nu1=0.5*(3.0*k1-2.0*mu1)/(3.0*k1+mu1)
      nu2=0.5*(3.0*k2-2.0*mu2)/(3.0*k2+mu2)
      c1=1.0-c2
c     Initialize the modulus tensors
      do 10 i=1,6
       do 20 j=1,6
        L1(i,j)=0.0
        L2(i,j)=0.0
20     continue
10    continue
c     Obtain the (6x6) modulus tensors, L1 and L2
      L1(1,1)=k1+(4.0/3.0)*mu1
      L1(2,2)=k1+(4.0/3.0)*mu1
      L1(3,3)=k1+(4.0/3.0)*mu1
      L1(1,2)=k1-(2.0/3.0)*mu1
      L1(2,1)=k1-(2.0/3.0)*mu1
      L1(2,3)=k1-(2.0/3.0)*mu1
      L1(3,2)=k1-(2.0/3.0)*mu1
      L1(3,1)=k1-(2.0/3.0)*mu1
      L1(1,3)=k1-(2.0/3.0)*mu1
      L1(4,4)=2.0*mu1
      L1(5,5)=2.0*mu1
      L1(6,6)=2.0*mu1
      L2(1,1)=k2+(4.0/3.0)*mu2
      L2(2,2)=k2+(4.0/3.0)*mu2
      L2(3,3)=k2+(4.0/3.0)*mu2
      L2(1,2)=k2-(2.0/3.0)*mu2
      L2(2,1)=k2-(2.0/3.0)*mu2
      L2(2,3)=k2-(2.0/3.0)*mu2
      L2(3,2)=k2-(2.0/3.0)*mu2
      L2(3,1)=k2-(2.0/3.0)*mu2
      L2(1,3)=k2-(2.0/3.0)*mu2
```

```fortran
      L2(4,4)=2.0*mu2
      L2(5,5)=2.0*mu2
      L2(6,6)=2.0*mu2
c     Begin calculating LHS
      do 21 i=1,6
       do 22 j=1,6
        d1(i,j)=L1(i,j)
22     continue
21    continue
      if(debug.eq..true.) then
        write(15,*)'Meff2'
        call flush(15)
      endif
      call inverse(d1,M1)
      call tenprod(M1,L2,d)
      if(debug.eq..true.) then
        write(15,*)'Meff3'
        call flush(15)
      endif
      do 30 i=1,6
       d(i,i)=d(i,i)-1.0
30    continue
      do 31 i=1,6
       do 32 j=1,6
        d1(i,j)=0.0
32     continue
31    continue
      call inverse(d,d1)
      if(debug.eq..true.) then
        write(15,*)'Meff4'
        call flush(15)
      endif
c     calculate S tensors
      if(debug.eq..true.) then
        write(15,*)'Meff5',a,b,c
        call flush(15)
      endif
      call stensor(a,b,c,nu1,k1,mu1,Si,pi1212,pi1313,pi2323)
      if(debug.eq..true.) then
        write(15,*)'Meff6'
        call flush(15)
      endif
      if(debug.eq..true.) then
        write(15,*)'Meff6a'
        call flush(15)
      endif
      do 40 i=1,6
       do 50 j=1,6
        d(i,j)=0.0
50     continue
40    continue
      if(debug.eq..true.) then
        write(15,*)'Meff6a_1',Si
        call flush(15)
      endif
      do 60 i=1,6
       do 70 j=1,6
      if(debug.eq..true.) then
        write(15,*)'j=',j,'   i=',i,Si(i,j)
        call flush(15)
      endif
        d(i,j)=Si(i,j)-c2*Si(i,j)
70     continue
60    continue
      if(debug.eq..true.) then
        write(15,*)'Meff6b'
        call flush(15)
      endif
      do 80 i=1,6
       do 90 j=1,6
        d(i,j)=d1(i,j)+d(i,j)
90     continue
80    continue
      if(debug.eq..true.) then
        write(15,*)'Meff6c'
        call flush(15)
      endif
      do 100 i=1,6
       do 110 j=1,6
        d1(i,j)=0.0
110    continue
100   continue
      if(debug.eq..true.) then
        write(15,*)'Meff7'
```

```
          call flush(15)
        endif
        call inverse(d,d1)
        if(debug.eq..true.) then
          write(15,*)'Meff7a'
          call flush(15)
        endif
        do 120 i=1,6
         do 130 j=1,6
          d(i,j)=c2*d1(i,j)
 130     continue
 120    continue
        do 140 i=1,6
         d(i,i)=d(i,i)+1.0
 140    continue
        do 150 i=1,6
         do 160 j=1,6
          d1(i,j)=0.0
 160     continue
 150    continue
        call tenprod(L1,d,LHS)
        do 161 i=1,6
         do 162 j=1,6
          d(i,j)=0.0
          d(i,j)=LHS(i,j)
 162     continue
 161    continue
        call inverse(d,MHS)
        if (path.eq..false.) then
         do 170 i=1,6
          do 180 j=1,6
           MHS(i,j)=LHS(i,j)
 180      continue
 170     continue
        endif
c----- symmetrize M to avoid small numerical variations
        do 190 i=1,6
         do 200 j=1,6
          M1(i,j)=MHS(i,j)
 200     continue
 190    continue
        do 210 i=1,6
         do 220 j=1,6
          MHS(i,j)=0.5*(M1(i,j)+M1(j,i))
 220     continue
 210    continue
        if(debug.eq..true.) then
          write(15,*)'Meff9'
          call flush(15)
        endif
        return
        end

c==========================================================================

c------ This is to evaluate the S tensor for an ellipsoidal inclusion.
          subroutine stensor(aold,bold,cold,nu,k1,mu1,s,p1212,p1313,p2323)
          real*8 ia,ib,ic,iaa,ibb,icc,iab,ibc,ica,iac,icb,iba
          real*8 a,b,c,p,el2
          real*8 ff,e
          real*8 s1111,s2222,s3333,s1122,s1133,s2233,s2211,s3311,s3322
          real*8 s1212,s1313,s2323
          real*8 q,r,nu,k1,mu1
          real*8 x,qqc,qqc2,aa,bb
          real*8 s(6,6)
          real*8 a1,b1,c1,s1(6,6),aold,bold,cold
          real*8 p1212,p1313,p2323,pp1212,pp1313,pp2323
          real*8 p1221,p1331,p2332,pp1221,pp1331,pp2332
          real*8 p2112,p3113,p3223,pp2112,pp3113,pp3223
          real*8 p2121,p3131,p3232,pp2121,pp3131,pp3232
          integer ip,i,j
          logical spath,debug
          common /paths/spath
         common /mdebug/debug
          p=3.14159265358979
          p1212=0.0
          p1313=0.0
          p2323=0.0
c         p=3.14159265
         if (dabs(aold-1.D0).lt.0.001) then
          a=1.D0
           else
          a=aold
           endif
```

```
      if (dabs(bold-1.D0).lt.0.001) then
       b=1.D0
        else
       b=bold
        endif
      if (dabs(cold-1.D0).lt.0.001) then
       c=1.D0
        else
       c=cold
        endif
      do i=1,6
       do j=1,6
        s(i,j)=0.0
        s1(i,j)=0.0
         enddo
        enddo
        ip=0
        ia=0.0
        ib=0.0
        ic=0.0
        iab=0.0
        ibc=0.0
        iac=0.0
        iaa=0.0
        ibb=0.0
        icc=0.0
      if ((a.eq.b).and.(b.eq.c)) then
         s1111=(8.D0*mu1+9.D0*k1)/(5.D0*(4.D0*mu1+3.D0*k1))
         s2222=(8.D0*mu1+9.D0*k1)/(5.D0*(4.D0*mu1+3.D0*k1))
         s3333=(8.D0*mu1+9.D0*k1)/(5.D0*(4.D0*mu1+3.D0*k1))
         s1122=(3.D0*k1-4.D0*mu1)/(5.D0*(4.D0*mu1+3.D0*k1))
         s1133=(3.D0*k1-4.D0*mu1)/(5.D0*(4.D0*mu1+3.D0*k1))
         s2233=(3.D0*k1-4.D0*mu1)/(5.D0*(4.D0*mu1+3.D0*k1))
         s2211=(3.D0*k1-4.D0*mu1)/(5.D0*(4.D0*mu1+3.D0*k1))
         s3311=(3.D0*k1-4.D0*mu1)/(5.D0*(4.D0*mu1+3.D0*k1))
         s3322=(3.D0*k1-4.D0*mu1)/(5.D0*(4.D0*mu1+3.D0*k1))
         s1212=(3.D0*(2.D0*mu1+k1))/(5.D0*(4.D0*mu1+3.D0*k1))
         s1313=(3.D0*(2.D0*mu1+k1))/(5.D0*(4.D0*mu1+3.D0*k1))
         s2323=(3.D0*(2.D0*mu1+k1))/(5.D0*(4.D0*mu1+3.D0*k1))
       go to 99
        endif
        if (((a.gt.c).and.(c.gt.b)).or.((a.eq.c).and.(c.gt.b))) then
c         interchange 2 and 3
          a1=a
          b1=b
          c1=c
          b=c
          c=b1
          ip=1
         endif
         if (((b.gt.a).and.(a.gt.c)).or.((a.eq.c).and.(c.lt.b))) then
c         interchange 1 and 2
          a1=a
          b1=b
          c1=c
          a=b
          b=a1
          ip=2
         endif
         if (((b.gt.c).and.(c.gt.a)).or.((b.eq.c).and.(b.gt.a))) then
c         make 2->1, 3->2, 1->3
          a1=a
          b1=b
          c1=c
          a=b
          b=c
          c=a1
          ip=3
         endif
         if ((c.gt.b).and.(b.gt.a)) then
c         interchange 1 and 3
          a1=a
          b1=b
          c1=c
          a=c
          c=a1
          ip=4
         endif
         if (((c.gt.a).and.(a.gt.b)).or.((a.eq.b).and.(b.lt.c))) then
c        make 3->1, 1->2, 2->3
          a1=a
          b1=b
          c1=c
          a=c
```

```
          b=a1
          c=b1
          ip=5
         endif
        if ((a.eq.b).and.(b.gt.c)) then
       call flush(15)
         ia=2.D0*p*a*a*c/(a*a-c*c)
         ia=ia/dsqrt(a*a-c*c)
         ia=ia*(dacos(c/a)-((c/a)*dsqrt(1.D0-((c/a)**2))))
         ib=ia
         ic=4.D0*p-ia-ib
         ibc=(ic-ib)/(3.D0*(b*b-c*c))
         iac=(ic-ia)/(3.D0*(a*a-c*c))
         iab=0.25*(((4.D0*p)/(3.D0*a*a))-iac)
         iba=iab
         ica=iac
         icb=ibc
         iaa=3.D0*iab
         ibb=((4.D0*p)/(3.D0*b*b))-iba-ibc
         icc=((4.D0*p)/(3.D0*c*c))-ica-icb
       go to 98
        endif
        if ((b.eq.c).and.(a.gt.b)) then
       call flush(15)
         ib=2.D0*p*a*c*c/(a*a-c*c)
         ib=ib/dsqrt(a*a-c*c)
         ib=ib*(((a/c)*dsqrt(((a/c)**2)-1.D0))-
     &             dlog((a/c)+dsqrt((a/c)**2-1.D0)))
         ic=ib
         ia=4.D0*p-ib-ic
         iab=(ib-ia)/(3.D0*(a*a-b*b))
         iac=(ic-ia)/(3.D0*(a*a-c*c))
         ibc=0.25*(((4.D0*p)/(3.D0*b*b))-iab)
         iba=iab
         ica=iac
         icb=ibc
         iaa=((4.D0*p)/(3.D0*a*a))-iab-iac
         icc=((4.D0*p)/(3.D0*c*c))-ica-icb
         ibb=3.D0*ibc
       go to 98
        endif
c------- input value of a;b and c
        if ((a.gt.b).and.(b.gt.c)) then
         ia=4.D0*p*a*b*c/(a*a-b*b)
         ia=ia/dsqrt((a*a-c*c))
         x=dsqrt(((a/c)**2)-1.D0)
         qqc=b*b-c*c
         qqc=qqc/(a*a-c*c)
         qqc=dsqrt(qqc)
         aa=1.D0
         bb=1.D0
         ff=el2(x,qqc,aa,bb)
         qqc2=qqc**2
         aa=1.D0
         bb=1.D0
         e=el2(x,qqc,aa,qqc2)
         ia=ia*(ff-e)
         ic=4.D0*p*a*b*c/(b*b-c*c)
         ic=ic/(dsqrt(a*a-c*c))
         ic=ic*((b*(dsqrt(a*a-c*c))/(a*c))-e)
         ib=4.D0*p-ia-ic
         iab=(ib-ia)/(3.D0*(a*a-b*b))
         ibc=(ic-ib)/(3.D0*(b*b-c*c))
         iac=(ic-ia)/(3.D0*(a*a-c*c))
         iba=iab
         ica=iac
         icb=ibc
         iaa=((4.D0*p)/(3.D0*a*a))-iab-iac
         ibb=((4.D0*p)/(3.D0*b*b))-iba-ibc
         icc=((4.D0*p)/(3.D0*c*c))-ica-icb
        endif
c------ elements of S
98      q=3.D0/(8.D0*p)
        q=q/(1.D0-nu)
        r=1.D0-2.D0*nu
        r=r/(8.D0*p*(1.D0-nu))
        s1111=q*a*a*iaa+r*ia
        s2222=q*b*b*ibb+r*ib
        s3333=q*c*c*icc+r*ic
        s1122=q*b*b*iab-r*ia
        s1133=q*c*c*iac-r*ia
        s2233=q*c*c*ibc-r*ib
        s2211=q*a*a*iba-r*ib
        s3311=q*a*a*ica-r*ic
```

```
      s3322=q*b*b*icb-r*ic
      s1212=0.5*q*(a*a+b*b)*iab+0.5*r*(ia+ib)
      s1313=0.5*q*(a*a+c*c)*iac+0.5*r*(ia+ic)
      s2323=0.5*q*(b*b+c*c)*ibc+0.5*r*(ib+ic)
99    continue
      if (spath.eq..true.) then
      s(1,1)=s1111
      s(1,2)=s1122
      s(1,3)=s1133
      s(2,1)=s2211
      s(2,2)=s2222
      s(2,3)=s2233
      s(3,1)=s3311
      s(3,2)=s3322
      s(3,3)=s3333
      s(4,4)=2.D0*s1212
      s(5,5)=2.D0*s2323
      s(6,6)=2.D0*s1313
      do 100 i=1,6
       do 110 j=1,6
        s1(i,j)=s(i,j)
110    continue
100   continue
      if (ip.eq.1) then
       s(1,2)=s1(1,3)
       s(1,3)=s1(1,2)
       s(2,1)=s1(3,1)
       s(2,2)=s1(3,3)
       s(2,3)=s1(3,2)
       s(3,1)=s1(2,1)
       s(3,2)=s1(2,3)
       s(3,3)=s1(2,2)
       s(4,4)=s1(6,6)
       s(6,6)=s1(4,4)
      else if (ip.eq.2) then
       s(1,1)=s1(2,2)
       s(1,2)=s1(2,1)
       s(1,3)=s1(2,3)
       s(2,1)=s1(1,2)
       s(2,2)=s1(1,1)
       s(2,3)=s1(1,3)
       s(3,1)=s1(3,2)
       s(3,2)=s1(3,1)
       s(5,5)=s1(6,6)
       s(6,6)=s1(5,5)
      else if (ip.eq.3) then
       s(1,1)=s1(3,3)
       s(1,2)=s1(3,1)
       s(1,3)=s1(3,2)
       s(2,1)=s1(1,3)
       s(2,2)=s1(1,1)
       s(2,3)=s1(1,2)
       s(3,1)=s1(2,3)
       s(3,2)=s1(2,1)
       s(3,3)=s1(2,2)
       s(4,4)=s1(6,6)
       s(5,5)=s1(4,4)
       s(6,6)=s1(5,5)
      else if (ip.eq.4) then
       s(1,1)=s1(3,3)
       s(1,2)=s1(3,2)
       s(1,3)=s1(3,1)
       s(2,1)=s1(2,3)
       s(2,3)=s1(2,1)
       s(3,1)=s1(1,3)
       s(3,2)=s1(1,2)
       s(3,3)=s1(1,1)
       s(4,4)=s1(5,5)
       s(5,5)=s1(4,4)
      else if (ip.eq.5) then
       s(1,1)=s1(2,2)
       s(1,2)=s1(2,3)
       s(1,3)=s1(2,1)
       s(2,1)=s1(3,2)
       s(2,2)=s1(3,3)
       s(2,3)=s1(3,1)
       s(3,1)=s1(1,2)
       s(3,2)=s1(1,3)
       s(3,3)=s1(1,1)
       s(4,4)=s1(5,5)
       s(5,5)=s1(6,6)
       s(6,6)=s1(4,4)
      endif
     else
```

```
c------ elements of Pi
        q=3.D0/(8.D0*p)
        q=q/(1.D0-nu)
        r=1.D0-2.D0*nu
        r=r/(8.D0*p*(1.D0-nu))
        P1313=(ic-ia)/(8.D0*p)
        P1331=(ic-ia)/(8.D0*p)
        P3131=(ia-ic)/(8.D0*p)
        P3113=(ia-ic)/(8.D0*p)
        P1212=(ib-ia)/(8.D0*p)
        P1221=(ib-ia)/(8.D0*p)
        P2121=(ia-ib)/(8.D0*p)
        P2112=(ia-ib)/(8.D0*p)
        P3232=(ib-ic)/(8.D0*p)
        P3223=(ib-ic)/(8.D0*p)
        P2323=(ic-ib)/(8.D0*p)
        P2332=(ic-ib)/(8.D0*p)
        pp1212=p1212
        pp1313=p1313
        pp2323=p2323
        pp1221=-pp1212
        pp2121=pp1221
        pp2112=-pp2121
        pp1331=-pp1313
        pp3131=pp1331
        pp3113=-pp3131
        pp2332=-pp2323
        pp3232=pp2332
        pp3223=-pp3232
        if (ip.eq.1) then
         p1212=pp1313
         p1313=pp1212
         p2323=pp3232
        else if (ip.eq.2) then
         p1212=pp2121
         p1313=pp2323
         p2323=pp1313
        else if (ip.eq.3) then
         p1212=pp3131
         p1313=pp3232
         p2323=pp1212
        else if (ip.eq.4) then
         p1212=pp3232
         p1313=pp3131
         p2323=pp2121
        else if (ip.eq.5) then
         p1212=pp2323
         p1313=pp2121
         p2323=pp3131
        endif
       endif
       if((ip.eq.1).or.(ip.eq.2).or.(ip.eq.3).or.(ip.eq.4).or.(ip.eq.5))
     &  then
         a=a1
         b=b1
         c=c1
        endif
        return
        end

      double precision FUNCTION EL2(X,QQC,AA,BB)
c     implicit real*8 (a-h,o-z)
      real*8 PI,ca,cb,x,qqc,aa,bb,qc,a,b,c,d,p
      real*8 z,eye,y,ff,em,e,g
      integer l
      PARAMETER(PI=3.14159265358979, CA=1.0D-6, CB=1.0D-12)
c     PARAMETER(PI=3.14159265, CA=1.0D-6, CB=1.0D-12)
      IF(X.EQ.0.0)THEN
        EL2=0.
      ELSE IF(QQC.NE.0.0)THEN
        QC=QQC
        A=AA
        B=BB
        C=X**2
        D=1.D0+C
        P=DSQRT((1.D0+QC**2*C)/D)
        D=X/D
        C=D/(2.*P)
        Z=A-B
        EYE=A
        A=0.5*(B+A)
        Y=DABS(1.D0/X)
        FF=0.
        L=0
```

```
          EM=1.D0
          QC=DABS(QC)
1         B=EYE*QC+B
          E=EM*QC
          G=E/P
          D=FF*G+D
          FF=C
          EYE=A
          P=G+P
          C=0.5*(D/P+C)
          G=EM
          EM=QC+EM
          A=0.5*(B/EM+A)
          Y=-E/Y+Y
          IF(Y.EQ.0.)Y=DSQRT(E)*CB
          IF(DABS(G-QC).GT.CA*G)THEN
            QC=DSQRT(E)*2.
            L=L+L
            IF(Y.LT.0.)L=L+1
            GO TO 1
          ENDIF
          IF(Y.LT.0.)L=L+1
          E=(DATAN(EM/Y)+PI*L)*A/EM
          IF(X.LT.0.)E=-E
          EL2=E+C*Z
        ELSE
          PAUSE 'failure in EL2'
        ENDIF
        RETURN
        END


C     This is a routine to invert matrices (6x6)
      subroutine inverse(a,y)
      real*8 a(6,6),y(6,6),d
      integer indx(6),i,j
      do 10 i=1,6
       do 11 j=1,6
        if (i.eq.j) then
         y(i,j)=1.0
        else
         y(i,j)=0.0
        endif
11     continue
10    continue
      call ludcmp(a,6,6,indx,d)
      do 13 j=1,6
       call lubksb(a,6,6,indx,y(1,j))
13    continue
      return
      end
c===============================================================================

C     This is a routine to invert matrices (3x3)
      subroutine inverse3x3(a,y)
      real*8 a(3,3),y(3,3),d
      integer indx(3),i,j
      do 10 i=1,3
       do 11 j=1,3
        if (i.eq.j) then
         y(i,j)=1.0
        else
         y(i,j)=0.0
        endif
11     continue
10    continue
      call ludcmp(a,3,3,indx,d)
      do 13 j=1,3
       call lubksb(a,3,3,indx,y(1,j))
13    continue
      return
      end
c===============================================================================


      SUBROUTINE LUBKSB(A,N,NP,INDX,B)
      implicit real*8 (a-h, o-z)
      implicit integer (i-n)
      real*8  A(NP,NP),B(N)
      integer indx(n)
      II=0
      DO 12 I=1,N
```

```
        LL=INDX(I)
        SUM=B(LL)
        B(LL)=B(I)
        IF (II.NE.0)THEN
          DO 11 J=II,I-1
            SUM=SUM-A(I,J)*B(J)
11        CONTINUE
        ELSE IF (SUM.NE.0.) THEN
          II=I
        ENDIF
        B(I)=SUM
12    CONTINUE
      DO 14 I=N,1,-1
        SUM=B(I)
        IF(I.LT.N)THEN
          DO 13 J=I+1,N
            SUM=SUM-A(I,J)*B(J)
13        CONTINUE
        ENDIF
        B(I)=SUM/A(I,I)
14    CONTINUE
      RETURN
      END


      SUBROUTINE LUDCMP(A,N,NP,INDX,D)
      integer indx(n),nmax,np,n,i,j,k,imax
      real*8 tiny,d
      PARAMETER (NMAX=100,TINY=1.0E-20)
      real*8 A(NP,NP),VV(NMAX),aamax,sum,dum
      D=1.
      DO 12 I=1,N
        AAMAX=0.
        DO 11 J=1,N
          IF (DABS(A(I,J)).GT.AAMAX) AAMAX=DABS(A(I,J))
11      CONTINUE
        IF (AAMAX.EQ.0.) PAUSE 'Singular matrix.'
        VV(I)=1./AAMAX
12    CONTINUE
      DO 19 J=1,N
        IF (J.GT.1) THEN
          DO 14 I=1,J-1
            SUM=A(I,J)
            IF (I.GT.1)THEN
              DO 13 K=1,I-1
                SUM=SUM-A(I,K)*A(K,J)
13            CONTINUE
              A(I,J)=SUM
            ENDIF
14        CONTINUE
        ENDIF
        AAMAX=0.
        DO 16 I=J,N
          SUM=A(I,J)
          IF (J.GT.1)THEN
            DO 15 K=1,J-1
              SUM=SUM-A(I,K)*A(K,J)
15          CONTINUE
            A(I,J)=SUM
          ENDIF
          DUM=VV(I)*DABS(SUM)
          IF (DUM.GE.AAMAX) THEN
            IMAX=I
            AAMAX=DUM
          ENDIF
16      CONTINUE
        IF (J.NE.IMAX)THEN
          DO 17 K=1,N
            DUM=A(IMAX,K)
            A(IMAX,K)=A(J,K)
            A(J,K)=DUM
17        CONTINUE
          D=-D
          VV(IMAX)=VV(J)
        ENDIF
        INDX(J)=IMAX
        IF(J.NE.N)THEN
          IF(A(J,J).EQ.0.)A(J,J)=TINY
          DUM=1./A(J,J)
          DO 18 I=J+1,N
            A(I,J)=A(I,J)*DUM
18        CONTINUE
        ENDIF
19    CONTINUE
```

```
      IF(A(N,N).EQ.0.)A(N,N)=TINY
      RETURN
      END

C      This is a routine to multiply 6x6 matrices (remember 4th order tensors
c      maybe written as 6x6 matrices without losing the tenosrial properties).
       subroutine tenprod(a,b,c)
       real*8 a(6,6), b(6,6), c(6,6)
       integer i,j,k
       do 20 i=1,6
        do 30 j=1,6
         c(i,j)=0.0
30      continue
20     continue
       do 10 i=1,6
        do 11 j=1,6
         do 12 k=1,6
          c(i,j)=a(i,k)*b(k,j)+c(i,j)
12       continue
11      continue
10     continue
       return
       end

c=============================================================================

       subroutine tenmatprod(m,s,d)
       real*8 m(6,6),s(6),d(6)
       integer i,j
       do 5 i=1,6
        d(i)=0.0
5      continue
       do 10 i=1,6
        do 20 j=1,6
         d(i)=m(i,j)*s(j)+d(i)
20      continue
10     continue
       return
       end
c=============================================================================

       subroutine rowcolumnprod(p,q,r)
       real*8 p(6),q(6),r
       integer i
       r=0.0
       do 5 i=1,6
        r=r+p(i)*q(i)
5      continue
       return
       end
c=============================================================================


c----- This is to calculate a part of the 'B' tensors for the composite
      subroutine Btensor(a,b,c,ad,bd,cd,c2,mu1,k1,mu2,k2,Bmat)
      real*8 Bmat(6,6),L1(6,6),L2(6,6),Si(6,6)
      real*8 M1(6,6),mu1,mu2,k1,k2,nu1,nu2
      real*8 a,b,c,ad,bd,cd,c1,c2
      real*8 d(6,6),d1(6,6)
      real*8 pi1212,pi2323,pi1313
      integer i,j
      logical spath,debug
      common /paths/spath
      common /mdebug/debug
      nu1=0.5*(3.0*k1-2.0*mu1)/(3.0*k1+mu1)
      nu2=0.5*(3.0*k2-2.0*mu2)/(3.0*k2+mu2)
      c1=1.0-c2
c     Initialize the modulus tensors
      do 10 i=1,6
       do 20 j=1,6
        L1(i,j)=0.0
        L2(i,j)=0.0
20     continue
10    continue
c     Obtain the (6x6) modulus tensors, L1 and L2
      L1(1,1)=k1+(4.0/3.0)*mu1
      L1(2,2)=k1+(4.0/3.0)*mu1
      L1(3,3)=k1+(4.0/3.0)*mu1
      L1(1,2)=k1-(2.0/3.0)*mu1
      L1(2,1)=k1-(2.0/3.0)*mu1
      L1(2,3)=k1-(2.0/3.0)*mu1
      L1(3,2)=k1-(2.0/3.0)*mu1
```

```fortran
      L1(3,1)=k1-(2.0/3.0)*mu1
      L1(1,3)=k1-(2.0/3.0)*mu1
      L1(4,4)=2.0*mu1
      L1(5,5)=2.0*mu1
      L1(6,6)=2.0*mu1
      L2(1,1)=k2+(4.0/3.0)*mu2
      L2(2,2)=k2+(4.0/3.0)*mu2
      L2(3,3)=k2+(4.0/3.0)*mu2
      L2(1,2)=k2-(2.0/3.0)*mu2
      L2(2,1)=k2-(2.0/3.0)*mu2
      L2(2,3)=k2-(2.0/3.0)*mu2
      L2(3,2)=k2-(2.0/3.0)*mu2
      L2(3,1)=k2-(2.0/3.0)*mu2
      L2(1,3)=k2-(2.0/3.0)*mu2
      L2(4,4)=2.0*mu2
      L2(5,5)=2.0*mu2
      L2(6,6)=2.0*mu2
c     Begin calculating B tensor
      call inverse(L1,M1)
      call tenprod(M1,L2,d)
      do 30 i=1,6
       d(i,i)=d(i,i)-1.0
30    continue
      call inverse(d,d1)
      do 31 i=1,6
       do 32 j=1,6
        d(i,j)=0.0
32     continue
31    continue
c     calculate S tensors
      spath=.true.
      call stensor(a,b,c,nu1,k1,mu1,Si,pi1212,pi1313,pi2323)
      do 60 i=1,6
       do 70 j=1,6
        d(i,j)=Si(i,j)-c2*Si(i,j)
70     continue
60    continue
      do 80 i=1,6
       do 90 j=1,6
        d(i,j)=d(i,j)+d1(i,j)
90     continue
80    continue
      call inverse(d,Bmat)
      return
      end


      doubleprecision FUNCTION RTSAFE(FUNCD,X1,X2,XACC)
      integer maxit,j
      real*8 x1,x2,xacc,fl,df,fh,xl,xh,swap,dxold,dx,f,temp
      logical neg,yield,check,cal
      common /mkdata2/yield,check,neg
      common /call/cal
      external funcd
      PARAMETER (MAXIT=100)
      CALL FUNCD(X1,FL,DF)
      if (neg.eq..true.) then
       write(15,*)'RTSAFE1'
       call flush(15)
       return
      endif
      CALL FUNCD(X2,FH,DF)
      if (neg.eq..true.) then
       write(15,*)'RTSAFE2'
       call flush(15)
       return
      endif
      IF(FL*FH.GE.0.) THEN
c      write(15,*)'data', x1,x2,fl,fh
c      call flush(15)
       PAUSE 'root must be bracketed'
      ENDIF
      IF(FL.LT.0.)THEN
        XL=X1
        XH=X2
      ELSE
        XH=X1
        XL=X2
        SWAP=FL
        FL=FH
        FH=SWAP
      ENDIF
      RTSAFE=.5*(X1+X2)
      DXOLD=DABS(X2-X1)
```

```
      DX=DXOLD
      CALL FUNCD(RTSAFE,F,DF)
      if (neg.eq..true.) then
       write(15,*)'RTSAFE2'
       call flush(15)
       return
      endif
      DO 11 J=1,MAXIT
        IF(((RTSAFE-XH)*DF-F)*((RTSAFE-XL)*DF-F).GE.0.
     *        .OR. DABS(2.D0*F).GT.ABS(DXOLD*DF) ) THEN
          DXOLD=DX
          DX=0.5*(XH-XL)
          RTSAFE=XL+DX
          IF(XL.EQ.RTSAFE) then
         if (cal.eq..true.) then
           write(15,*)'ITER',J
         endif
           RETURN
          endif
        ELSE
          DXOLD=DX
          DX=F/DF
          TEMP=RTSAFE
          RTSAFE=RTSAFE-DX
          IF(TEMP.EQ.RTSAFE) then
         if (cal.eq..true.) then
           write(15,*) 'ITER',J
         endif
           RETURN
          endif
        ENDIF
        IF(DABS(DX).LT.XACC) then
       if (cal.eq..true.) then
         write(15,*)'ITER',J
       call flush(15)
       endif
         RETURN
        endif
        CALL FUNCD(RTSAFE,F,DF)
      if (neg.eq..true.) then
       return
      endif
        IF(F.LT.0.) THEN
          XL=RTSAFE
          FL=F
        ELSE
          XH=RTSAFE
          FH=F
        ENDIF
11    CONTINUE
c     PAUSE 'RTSAFE exceeding maximum iterations'
      write(15,*)'RTSAFE NOT GOOD'
      call flush(15)
      neg=.true.
      RETURN
      END



c================================================================================

       doubleprecision function func(dlam)
       real*8 s(3,3),phi,MHS(6,6),Amat(6,6),Ce(6,6)
       real*8 sn(3,3),n(3,3),omega(3,3),sige(3,3)
       real*8 strainc(6),sigec(6),vec(3),R(3,3)
       real*8 f,wi1,wi2,sci(6),dumd(6,6),RT(3,3)
       real*8 fn,wi1n,wi2n,dlam,sigy1n,dinc(6)
       real*8 sc(6),nc(6),omegac(6),sig_n(6)
       real*8 mu1,k1,mu2,k2,sigy1,sigma_n(6)
       real*8 dummy(6),a,b,c,ad,bd,cd,dummy1(3,3),dummy2(3,3)
       real*8 dummy5(6),dum1,dum2,sn1(3,3),sigy0
       real*8 dep_plas,eps_plastic,deps(3,3),k11,eps_plasticn
       real*8 an,bn,cn,adn,bdn,cdn,e1212,e2323,e1313
       real*8 pi1212,pi2323,pi1313,bmat(6,6),nu1
       real*8 mu1e,k1e,mu2e,ek2e,y,MHSn(6,6),dum5,dum10
       real*8 cp,sp,ct,st,cs,ss,rot(3,3),rott(3,3),hrd
       real*8 athetan,apsin,aphin,atheta,aphi,apsi
       real*8 cpn,spn,ctn,stn,csn,ssn,rotn(3,3),rotnt(3,3)
       real*8 loadp(6),loading
       real*8 zero(6),totstrain(6)
       real*8 rotj(3,3),rotjn(3,3),rota(3,3)
       integer i,j,ninc
```

```
      logical path,yield,check,neg,spath,yc,debug
      logical evolf,evolwi,evolti
      common /controls/evolf,evolwi,evolti
      common /mkmodulus/mu1,k1,mu2,k2,sigy0,k11
      common /mkdata1/f,wi1,wi2,sc,strainc,eps_plastic,sigy1,hrd
      common /mkdata2/yield,check,neg
      common /mkelas/mu1e,k1e,mu2e,ek2e
      common /mkorient/cp,sp,ct,st,cs,ss,rot,rott,atheta,aphi,apsi
      common /mkrot/R,RT,rotj
      common /paths/spath
      common /mdebug/debug
c
      path=.true.
      yield=.true.
      neg=.false.
      nu1=0.5*(3.0*k1-2.0*mu1)/(3.0*k1+mu1)
     if(debug.eq..true.) then
     write(15,*)'func1'
     call flush(15)
     endif
c
      c=1.000
      a=c/wi1
      b=c/wi2
      ad=a
      bd=b
      cd=c
c
c------ calculate Meffective
      call Meffective(a,b,c,ad,bd,cd,f,mu1,k11,mu2,k2,MHS,path)
      do 100 i=1,6
       do 110 j=1,6
        MHS(i,j)=3.0*mu1*MHS(i,j)
110    continue
100   continue
c
c---------------------------------------------------------------------------
c------ Estimate stress
c------ STEP 1: Estimate the elastic predictor
      path=.false.
      call Meffective(a,b,c,ad,bd,cd,f,mu1e,k1e,mu2e,ek2e,Ce,path)
      path=.true.
      call tenmatprod(Ce,strainc,sigec)
     do 201 i=1,6
      loadp(i)=sigec(i)
201    continue
       sige(1,1)=sc(1)+sigec(1)
       sige(2,2)=sc(2)+sigec(2)
       sige(3,3)=sc(3)+sigec(3)
       sige(1,2)=(sc(4)+sigec(4))/dsqrt(2.D0)
       sige(2,3)=(sc(5)+sigec(5))/dsqrt(2.D0)
       sige(3,1)=(sc(6)+sigec(6))/dsqrt(2.D0)
       sige(2,1)=sige(1,2)
       sige(3,2)=sige(2,3)
       sige(1,3)=sige(3,1)
       sigec(1)=sige(1,1)
       sigec(2)=sige(2,2)
       sigec(3)=sige(3,3)
       sigec(4)=sige(1,2)*dsqrt(2.D0)
       sigec(5)=sige(2,3)*dsqrt(2.D0)
       sigec(6)=sige(3,1)*dsqrt(2.D0)
c------ STEP 1a: Determining the stress on the yield surface
       sigma_n(1)=sc(1)
       sigma_n(2)=sc(2)
       sigma_n(3)=sc(3)
       sigma_n(4)=sc(4)
       sigma_n(5)=sc(5)
       sigma_n(6)=sc(6)
       call yieldtest(MHS,sigma_n,sigy1,f,yc,y)
       if (yc.eq..false.) then
        call yieldtest(MHS,sigec,sigy1,f,yc,y)
        if (yc.eq..true.) then
c        this is the step where the behavior becomes plastic (the previous
c        step was elastic)
c        write(15,*)'Becoming plastic'
       do 127 i=1,6
        zero(i)=0.0
        totstrain(i)=strainc(i)
127      continue
         ninc=100.0
         call stre(sigma_n,sigec,zero,totstrain,sig_n,Ce,ninc)
        else
        write(15,*)'NOT PLASTIC'
        call flush(15)
```

```
         do 130 i=1,6
          sig_n(i)=sc(i)
130      continue
        endif
       else
        call flush(15)
        do 131 i=1,6
         sig_n(i)=sc(i)
131     continue
       endif
c------ calculate N(3,3)
      call tenmatprod(MHS,sig_n,nc)
       do 200 i=1,6
        nc(i)=nc(i)/(1.0-f)
        nc(i)=2.0*nc(i)/sigy1
        dummy(i)=nc(i)
        dummy5(i)=nc(i)
200    continue
       n(1,1)=nc(1)
       n(2,2)=nc(2)
       n(3,3)=nc(3)
       n(1,2)=nc(4)/dsqrt(2.D0)
       n(2,1)=n(1,2)
       n(2,3)=nc(5)/dsqrt(2.D0)
       n(3,2)=n(2,3)
       n(3,1)=nc(6)/dsqrt(2.D0)
       n(1,3)=n(3,1)
     call rowcolumnprod(loadp,dummy5,loading)
     if (loading.le.0.0) then
      write(15,*)'loading in func',loading
       endif
       do 202 i=1,6
        dummy5(i)=nc(i)
202    continue
c------ STEP 2: Remaining part of the stress
c------ STEP 2a: Calculating 'omega'
       spath=.false.
     do 17 i=1,6
      do 18 j=1,6
       dumd(i,j)=0.0
18      continue
17     continue
     if(debug.eq..true.) then
       write(15,*)'func3',a,b,c,nu1,k1,mu1,dumd,pi1212,pi1313,
    &                  pi2323
     call flush(15)
     endif
      call stensor(a,b,c,nu1,k1,mu1,dumd,pi1212,pi1313,pi2323)
      e1212=pi1212-f*pi1212
      e2323=pi2323-f*pi2323
      e1313=pi1313-f*pi1313
      spath=.true.
     if(debug.eq..true.) then
     write(15,*)'func3_a',ad,bd,cd,a,b,c
     call flush(15)
     endif
      call Btensor(a,b,c,ad,bd,cd,f,mu1,k1,mu2,k2,Bmat)
      do 41 i=1,3
       do 42 j=1,6
        Bmat(i,j)=0.0
42     continue
41    continue
      Bmat(4,1)=-2.0*e1212*Bmat(4,1)
      Bmat(4,2)=-2.0*e1212*Bmat(4,2)
      Bmat(4,3)=-2.0*e1212*Bmat(4,3)
      Bmat(4,4)=-2.0*e1212*Bmat(4,4)
      Bmat(4,5)=-2.0*e1212*Bmat(4,5)
      Bmat(4,6)=-2.0*e1212*Bmat(4,6)
      Bmat(5,1)=-2.0*e2323*Bmat(5,1)
      Bmat(5,2)=-2.0*e2323*Bmat(5,2)
      Bmat(5,3)=-2.0*e2323*Bmat(5,3)
      Bmat(5,4)=-2.0*e2323*Bmat(5,4)
      Bmat(5,5)=-2.0*e2323*Bmat(5,5)
      Bmat(5,6)=-2.0*e2323*Bmat(5,6)
      Bmat(6,1)=-2.0*e1313*Bmat(6,1)
      Bmat(6,2)=-2.0*e1313*Bmat(6,2)
      Bmat(6,3)=-2.0*e1313*Bmat(6,3)
      Bmat(6,4)=-2.0*e1313*Bmat(6,4)
      Bmat(6,5)=-2.0*e1313*Bmat(6,5)
      Bmat(6,6)=-2.0*e1313*Bmat(6,6)
     if(debug.eq..true.) then
     write(15,*)'func3_b',a,b,c,ad,bd,cd
     call flush(15)
     endif
```

```
       call Atensor(a,b,c,ad,bd,cd,f,mu1,k1,mu2,k2,Amat)
c-------
      if(debug.eq..true.) then
      write(15,*)'func4'
      call flush(15)
      endif
      do 3 i=1,3
       do 4 j=1,3
        omega(i,j)=0.0
4      continue
3     continue
      do 1 i=1,3
       omega(1,2)=-Bmat(4,i)*nc(i)+omega(1,2)
       omega(2,3)=-Bmat(5,i)*nc(i)+omega(2,3)
       omega(1,3)=-Bmat(6,i)*nc(i)+omega(1,3)
1     continue
      do 2 i=4,6
       omega(1,2)=-Bmat(4,i)*nc(i)+omega(1,2)
       omega(2,3)=-Bmat(5,i)*nc(i)+omega(2,3)
       omega(1,3)=-Bmat(6,i)*nc(i)+omega(1,3)
2     continue
      omega(2,1)=-omega(1,2)
      omega(3,2)=-omega(2,3)
      omega(3,1)=-omega(1,3)
      do 7 i=1,3
       do 8 j=1,3
        omega(i,j)=omega(i,j)/dsqrt(2.0d0)
8      continue
7     continue
      call tenmatprod(Amat,nc,dinc)
       if (dabs(a-b).gt.0.01) then
        omega(1,2)=omega(1,2)-(a*a+b*b)*dinc(4)/(dsqrt(2.0d0)*(a*a-b*b))
       endif
       if (dabs(a-c).gt.0.01) then
        omega(1,3)=omega(1,3)-(a*a+c*c)*dinc(6)/(dsqrt(2.0d0)*(a*a-c*c))
       endif
       if (dabs(c-b).gt.0.01) then
        omega(2,3)=omega(2,3)-(b*b+c*c)*dinc(5)/(dsqrt(2.0d0)*(b*b-c*c))
       endif
       omega(2,1)=-omega(1,2)
       omega(3,2)=-omega(2,3)
       omega(3,1)=-omega(1,3)
       omegac(1)=omega(1,1)
       omegac(2)=omega(2,2)
       omegac(3)=omega(3,3)
       omegac(4)=omega(1,2)*dsqrt(2.D0)
       omegac(5)=omega(2,3)*dsqrt(2.D0)
       omegac(6)=omega(1,3)*dsqrt(2.D0)
c-------------------
       s(1,1)=sc(1)
       s(2,2)=sc(2)
       s(3,3)=sc(3)
       s(1,2)=sc(4)/dsqrt(2.D0)
       s(2,3)=sc(5)/dsqrt(2.D0)
       s(3,1)=sc(6)/dsqrt(2.D0)
       s(2,1)=s(1,2)
       s(3,2)=s(2,3)
       s(1,3)=s(3,1)
c      write(15,*)'In func',sc
c      call flush(15)
       call matprod(s,omega,dummy1)
       call matprod(omega,s,dummy2)
       do 300 i=1,3
         do 310 j=1,3
         sn(i,j)=dummy1(i,j)-dummy2(i,j)
ccc         sn(i,j)=0.0
310      continue
300     continue
       do 450 i=1,6
        dummy5(i)=nc(i)
450    continue
      if(debug.eq..true.) then
      write(15,*)'func5'
      call flush(15)
      endif
        call tenmatprod(Ce,dummy5,dummy)
        sn(1,1)=sn(1,1)-dummy(1)
        sn(2,2)=sn(2,2)-dummy(2)
        sn(3,3)=sn(3,3)-dummy(3)
        sn(1,2)=sn(1,2)-dummy(4)/dsqrt(2.D0)
        sn(2,3)=sn(2,3)-dummy(5)/dsqrt(2.D0)
        sn(3,1)=sn(3,1)-dummy(6)/dsqrt(2.D0)
        sn(2,1)=sn(1,2)
        sn(3,2)=sn(2,3)
```

```
         sn(1,3)=sn(3,1)
         do 320 i=1,3
          do 330 j=1,3
           dummy1(i,j)=sn(i,j)
330       continue
320      continue
c------ STEP 3: put them together
         sn1(1,1)=sige(1,1)+dlam*sn(1,1)
         sn1(2,2)=sige(2,2)+dlam*sn(2,2)
         sn1(3,3)=sige(3,3)+dlam*sn(3,3)
         sn1(1,2)=sige(1,2)+dlam*sn(1,2)
         sn1(2,3)=sige(2,3)+dlam*sn(2,3)
         sn1(3,1)=sige(3,1)+dlam*sn(3,1)
         sn1(2,1)=sn1(1,2)
         sn1(3,2)=sn1(2,3)
         sn1(1,3)=sn1(3,1)
c------ These components of sn1 are now NOT in the coordinate farme
c------ coincides with the orientation of the inclusions. This is due
c------ the algorithm used (see notes for more details). In order to
c        obtain them in this coordinate frame:
         call matprod(R,sn1,dummy1)
         call matprod(dummy1,RT,sn1)
c        write(15,*)'sn1',sn1(3,3)
ccall flush(15)
c------------------------------------------------------------------------------
c----- estimate the new state variables for this iteration
       if(debug.eq..true.) then
       write(15,*)'func6'
       call flush(15)
       endif
        call Atensor(a,b,c,ad,bd,cd,f,mu1,k1,mu2,k2,Amat)
       if(debug.eq..true.) then
       write(15,*)'func6a'
       call flush(15)
       endif
        if (evolf.eq..true.) then
         fn=f+(n(1,1)+n(2,2)+n(3,3))*(1.0-f)*dlam
        else
         fn=f
        endif
        if ((fn.lt.0.0).or.(fn.gt.1.0)) then
       write(15,*)'porosity estimate wrong',f,dlam
       call flush(15)
c        neg=.true.
c        return
         f=0.0
        endif
        dummy(1)=Amat(3,1)-Amat(1,1)
        dummy(2)=Amat(3,2)-Amat(1,2)
        dummy(3)=Amat(3,3)-Amat(1,3)
        dummy(4)=Amat(3,4)-Amat(1,4)
        dummy(5)=Amat(3,5)-Amat(1,5)
        dummy(6)=Amat(3,6)-Amat(1,6)
        call rowcolumnprod(dummy,dummy5,dum1)
        do 400 i=1,6
         dummy5(i)=nc(i)
400     continue
        dummy(1)=Amat(3,1)-Amat(2,1)
        dummy(2)=Amat(3,2)-Amat(2,2)
        dummy(3)=Amat(3,3)-Amat(2,3)
        dummy(4)=Amat(3,4)-Amat(2,4)
        dummy(5)=Amat(3,5)-Amat(2,5)
        dummy(6)=Amat(3,6)-Amat(2,6)
        call rowcolumnprod(dummy,dummy5,dum2)
       if(debug.eq..true.) then
       write(15,*)'func6b'
       call flush(15)
       endif
        if (evolwi.eq..true.) then
         wi1n=wi1+dlam*wi1*dum1
         wi2n=wi2+dlam*wi2*dum2
        else
       wi1n=wi1
       wi2n=wi2
        endif
        dum5=dum1
        dum10=dum2
        if ((wi1n.le.0.0).or.(wi2n.le.0.0)) then
       write(15,*)'aspect estimate wrong','dlam=',dlam
       call flush(15)
         neg=.true.
         return
        endif
c----- estimating new orientations
```

```
          if (evolti.eq..true.) then
           athetan=atheta-(omega(2,3)*cp+omega(1,3)*sp)*dlam
         if (dabs(wi2n-1.0).lt.0.01) then
          athetan=0.0
           endif
           if (st.ne.0.0) then
            aphin=aphi-((omega(2,3)*ss/st)-omega(1,3)*cs/st)*dlam
            apsin=apsi+(-omega(1,2)+(ct/st)*(omega(2,3)*ss-omega(1,3)*cs))*
     &                dlam
           else
            apsin=0.0
            aphin=0.0
           endif
         call matprod(R,rotj,rotjn)
          else
           aphin=aphi
           apsin=apsi
           athetan=atheta
         do 27 i=1,3
          do 28 j=1,3
           rotjn(i,j)=rotj(i,j)
 28         continue
 27        continue
          endif
          ctn=dcos(athetan)
          stn=dsin(athetan)
          cpn=dcos(aphin)
          csn=dcos(apsin)
          spn=dsin(aphin)
          ssn=dsin(apsin)
          rota(1,1)=csn*cpn-ssn*ctn*spn
          rota(1,2)=-csn*spn-ssn*ctn*cpn
          rota(1,3)=ssn*stn
          rota(2,1)=ssn*cpn+csn*ctn*spn
          rota(2,2)=-ssn*spn+csn*ctn*cpn
          rota(2,3)=-csn*stn
          rota(3,1)=stn*spn
          rota(3,2)=stn*cpn
          rota(3,3)=ctn
          call matprod(rota,rotjn,rotn)
           do 77 i=1,3
            do 88 j=1,3
             rotnt(i,j)=rotn(j,i)
 88          continue
 77         continue
c----- estimating new sig_yield
         do 501 i=1,3
          do 502 j=1,3
           deps(i,j)=dlam*n(i,j)
 502      continue
 501     continue
         dum1=(deps(1,1)+deps(2,2)+deps(3,3))/3.0
         do 503 i=1,3
          deps(i,i)=deps(i,i)-dum1
 503     continue
         dep_plas=deps(1,1)**2+deps(2,2)**2+deps(3,3)**2
         dep_plas=dep_plas+2.0*(deps(1,2)**2+deps(2,3)**2+deps(3,1)**2)
         dep_plas=(2.0/3.0)*dep_plas
         eps_plasticn=eps_plastic+dsqrt(dep_plas)
ccc         sigy1n=sigy0*((1.0+eps_plasticn)**(1.0/3.0))
          sigy1n=sigy1
c-----------------------------------------------------------------------
c----- calculate the yield function with these new values of the stress
c      and state variables.....
c
c
        cn=1.000
        an=cn/wi1n
        bn=cn/wi2n
        adn=an
        bdn=bn
        cdn=cn
       call Meffective(an,bn,cn,adn,bdn,cdn,fn,mu1,k11,mu2,k2,MHSn,path)
       do 500 i=1,6
        do 510 j=1,6
         MHSn(i,j)=3.0*mu1*MHSn(i,j)
510     continue
500    continue
c----- The stress must now be expressed w.r.t the new basis -- the basis
c      which is aligned with the current (new) orientation of the voids. This is
c      the basis in which the components of MHSn are obtained.
c
c----- This expresses sn1 in GLOBAL frame.
        call matprod(rot,sn1,dummy)
```

```
        call matprod(dummy,rott,sn1)
c----- This expresses sn1 w.r.t new orientation of particles
        call matprod(rotnt,sn1,dummy)
        call matprod(dummy,rotn,sn1)
c
      sci(1)=sn1(1,1)
      sci(2)=sn1(2,2)
      sci(3)=sn1(3,3)
      sci(4)=sn1(1,2)*dsqrt(2.D0)
      sci(5)=sn1(2,3)*dsqrt(2.D0)
      sci(6)=sn1(3,1)*dsqrt(2.D0)
      call tenmatprod(MHSn,sci,dummy)
      call rowcolumnprod(sci,dummy,phi)
      phi=phi/((1.0-fn)*sigy1n)
      phi=phi-sigy1n
      func=phi
      return
      end


      doubleprecision FUNCTION RTNEWT(FUNCD,X2,XACC)
      integer jmax,j
c     implicit doubleprecision(a-h,o-z)
      real*8 x2,xacc,f,df,dx
      logical neg,check,yield
      common /mkdata2/yield,check,neg
      external funcd
      PARAMETER (JMAX=20)
      rtnewt=x2
      DO 11 J=1,JMAX
        CALL FUNCD(RTNEWT,F,DF)
      if (neg.eq..true.) then
       return
        endif
        DX=F/DF
        RTNEWT=RTNEWT-DX
c       RTNEWT=RTNEWT-0.000001
        IF(DABS(DX).LT.XACC) then
         RETURN
        endif
11    CONTINUE
c     PAUSE 'RTNEWT exceeding maximum iterations'
      write(15,*)'RTNEWT NOT GOOD'
      call flush(15)
      neg=.true.
      END


       subroutine guessmaker(dlam)
       real*8 MHS(6,6),Ce(6,6)
       real*8 n(3,3),sige(3,3),L
       real*8 strainc(6),sigec(6)
       real*8 f,wi1,wi2
       real*8 cp,sp,ct,st,cs,ss,rot(3,3),rott(3,3),atheta,aphi,apsi
       real*8 dlam,sigy0
       real*8 sc(6),nc(6),sig_n(6)
       real*8 mu1,k1,mu2,k2,sigy1,sigma_n(6)
       real*8 dummy(6),a,b,c,ad,bd,cd
       real*8 dummy5(6),k11,mu1e,k1e,mu2e,ek2e,eps_plastic
       real*8 phi1,ftest,phi2,dphidf,h,wi1test,ate,bte
       real*8 dphidwi1,wi2test,dphidwi2,dum5,dum10,y
       real*8 omega(3,3),omegac(6),dummy1(3,3),dummy2(3,3)
       real*8 bmat(6,6),amat(6,6),dumd(6,6),dinc(6),hrd
       real*8 pi1212,pi2323,pi1313,e1212,e2323,e1313
       real*8 nu1,zero(6),totstrain(6)
       integer i,j,ninc
       logical path,yield,check,neg,yc,spath,debug
       logical evolf,evolwi,evolti
       common /controls/evolf,evolwi,evolti
       common /mkmodulus/mu1,k1,mu2,k2,sigy0,k11
       common /mkelas/mu1e,k1e,mu2e,ek2e
       common /mkdata1/f,wi1,wi2,sc,strainc,eps_plastic,sigy1,hrd
       common /mkorient/cp,sp,ct,st,cs,ss,rot,rott,atheta,aphi,apsi
       common /mkdata2/yield,check,neg
       common /mdebug/debug

c
       path=.true.
       yield=.true.
       c=1.000
       a=c/wi1
       b=c/wi2
       ad=a
```

```
       bd=b
       cd=c
c
c-------------------------------------------------------------------------------
c------ Estimate stress
c------ STEP 1: Estimate the elastic predictor
       path=.false.
       call Meffective(a,b,c,ad,bd,cd,f,mu1e,k1e,mu2e,ek2e,Ce,path)
       path=.true.
       call tenmatprod(Ce,strainc,sigec)
        sige(1,1)=sc(1)+sigec(1)
        sige(2,2)=sc(2)+sigec(2)
        sige(3,3)=sc(3)+sigec(3)
        sige(1,2)=(sc(4)+sigec(4))/dsqrt(2.D0)
        sige(2,3)=(sc(5)+sigec(5))/dsqrt(2.D0)
        sige(3,1)=(sc(6)+sigec(6))/dsqrt(2.D0)
        sige(2,1)=sige(1,2)
        sige(3,2)=sige(2,3)
        sige(1,3)=sige(3,1)
        sigec(1)=sige(1,1)
        sigec(2)=sige(2,2)
        sigec(3)=sige(3,3)
        sigec(4)=sige(1,2)*dsqrt(2.D0)
        sigec(5)=sige(2,3)*dsqrt(2.D0)
        sigec(6)=sige(3,1)*dsqrt(2.D0)
c------ Test for yielding...
c------ calculate yield function with stress=elastic predictor
        call Meffective(a,b,c,ad,bd,cd,f,mu1,k1,mu2,k2,MHS,path)
        do 101 i=1,6
         do 111 j=1,6
          MHS(i,j)=3.0*mu1*MHS(i,j)
111      continue
101     continue
        call yieldtest(MHS,sigec,sigy1,f,yc,y)
        if (yc.eq..false.) then
         yield=.false.
         go to 999
        endif
       if(debug.eq..true.) then
       write(15,*)'guess2'
       call flush(15)
       endif
c------ STEP 1a: Determining the stress on the yield surface
        sigma_n(1)=sc(1)
        sigma_n(2)=sc(2)
        sigma_n(3)=sc(3)
        sigma_n(4)=sc(4)
        sigma_n(5)=sc(5)
        sigma_n(6)=sc(6)
        call yieldtest(MHS,sigma_n,sigy1,f,yc,y)
        if (yc.eq..false.) then
         call yieldtest(MHS,sigec,sigy1,f,yc,y)
         if (yc.eq..true.) then
c         this is the step where the behavior becomes plastic (the previous
c         step was elastic)
        do 127 i=1,6
         zero(i)=0.0
         totstrain(i)=strainc(i)
127       continue
          ninc=100.0
          call stre(sigma_n,sigec,zero,totstrain,sig_n,Ce,ninc)
         else
          do 130 i=1,6
           sig_n(i)=sc(i)
130       continue
         endif
        else
          do 131 i=1,6
           sig_n(i)=sc(i)
131       continue
        endif
c------ calculate N(3,3)
       call tenmatprod(MHS,sig_n,nc)
        do 200 i=1,6
         nc(i)=nc(i)/(1.0-f)
         nc(i)=2.0*nc(i)/sigy1
200     continue
        n(1,1)=nc(1)
        n(2,2)=nc(2)
        n(3,3)=nc(3)
        n(1,2)=nc(4)/dsqrt(2.D0)
        n(2,1)=n(1,2)
        n(2,3)=nc(5)/dsqrt(2.D0)
        n(3,2)=n(2,3)
```

```
        n(3,1)=nc(6)/dsqrt(2.D0)
        n(1,3)=n(3,1)
c------------------------------------------------------------------------------
c----- calculate guess
       do 410 i=1,6
        dummy5(i)=nc(i)
410    continue
       call tenmatprod(Ce,dummy5,dummy)
       do 420 i=1,6
        dummy5(i)=nc(i)
420    continue
       call rowcolumnprod(dummy,dummy5,L)
       if(debug.eq..true.) then
       write(15,*)'guess3'
       call flush(15)
       endif
c----- calculating omega
       spath=.false.
       nu1=0.5*(3.0*k1-2.0*mu1)/(3.0*k1+mu1)
       call stensor(a,b,c,nu1,k1,mu1,dumd,pi1212,pi1313,pi2323)
       e1212=pi1212-f*pi1212
       e2323=pi2323-f*pi2323
       e1313=pi1313-f*pi1313
       spath=.true.
       call Btensor(a,b,c,ad,bd,cd,f,mu1,k1,mu2,k2,Bmat)
       do 41 i=1,3
        do 42 j=1,6
         Bmat(i,j)=0.0
42      continue
41     continue
       Bmat(4,1)=2.0*e1212*Bmat(4,1)
       Bmat(4,2)=2.0*e1212*Bmat(4,2)
       Bmat(4,3)=2.0*e1212*Bmat(4,3)
       Bmat(4,4)=2.0*e1212*Bmat(4,4)
       Bmat(4,5)=2.0*e1212*Bmat(4,5)
       Bmat(4,6)=2.0*e1212*Bmat(4,6)
       Bmat(5,1)=2.0*e2323*Bmat(5,1)
       Bmat(5,2)=2.0*e2323*Bmat(5,2)
       Bmat(5,3)=2.0*e2323*Bmat(5,3)
       Bmat(5,4)=2.0*e2323*Bmat(5,4)
       Bmat(5,5)=2.0*e2323*Bmat(5,5)
       Bmat(5,6)=2.0*e2323*Bmat(5,6)
       Bmat(6,1)=2.0*e1313*Bmat(6,1)
       Bmat(6,2)=2.0*e1313*Bmat(6,2)
       Bmat(6,3)=2.0*e1313*Bmat(6,3)
       Bmat(6,4)=2.0*e1313*Bmat(6,4)
       Bmat(6,5)=2.0*e1313*Bmat(6,5)
       Bmat(6,6)=2.0*e1313*Bmat(6,6)
       call Atensor(a,b,c,ad,bd,cd,f,mu1,k1,mu2,k2,Amat)
       do 3 i=1,3
        do 4 j=1,3
         omega(i,j)=0.0
4       continue
3      continue
       do 1 i=1,3
        omega(1,2)=-Bmat(4,i)*nc(i)+omega(1,2)
        omega(2,3)=-Bmat(5,i)*nc(i)+omega(2,3)
        omega(1,3)=-Bmat(6,i)*nc(i)+omega(1,3)
1      continue
       do 2 i=4,6
        omega(1,2)=-Bmat(4,i)*nc(i)+omega(1,2)
        omega(2,3)=-Bmat(5,i)*nc(i)+omega(2,3)
        omega(1,3)=-Bmat(6,i)*nc(i)+omega(1,3)
2       continue
      omega(2,1)=-omega(1,2)
      omega(3,2)=-omega(2,3)
      omega(3,1)=-omega(1,3)
      do 7 i=1,3
       do 8 j=1,3
        omega(i,j)=omega(i,j)/dsqrt(2.0d0)
8      continue
7     continue
      call tenmatprod(Amat,nc,dinc)
       if (dabs(a-b).gt.0.01) then
       omega(1,2)=omega(1,2)-(a*a+b*b)*dinc(4)/(dsqrt(2.0d0)*(a*a-b*b))
       endif
       if (dabs(a-c).gt.0.01) then
       omega(1,3)=omega(1,3)-(a*a+c*c)*dinc(6)/(dsqrt(2.0d0)*(a*a-c*c))
       endif
       if (dabs(c-b).gt.0.01) then
       omega(2,3)=omega(2,3)-(b*b+c*c)*dinc(5)/(dsqrt(2.0d0)*(b*b-c*c))
       endif
       omega(2,1)=-omega(1,2)
       omega(3,2)=-omega(2,3)
```

```
      omega(3,1)=-omega(1,3)
      omegac(1)=omega(1,1)
      omegac(2)=omega(2,2)
      omegac(3)=omega(3,3)
      omegac(4)=omega(1,2)*dsqrt(2.D0)
      omegac(5)=omega(2,3)*dsqrt(2.D0)
      omegac(6)=omega(1,3)*dsqrt(2.D0)
      call matprod(sig_n,omega,dummy1)
      call matprod(omega,sig_n,dummy2)
      do 12 i=1,3
       do 13 j=1,3
        dummy1(i,j)=dummy1(i,j)-dummy2(i,j)
13     continue
12    continue
      do 14, i=1,3
       do 15 j=1,3
        L=L-dummy1(i,j)*n(i,j)
15     continue
14    continue
c----- calculating H
      if(debug.eq..true.) then
      write(15,*)'guess4'
      call flush(15)
      endif
c----- step A -- calculating dphi/df
       H=0.0
     if (evolf.eq..true.) then
       path=.true.
       call Meffective(a,b,c,ad,bd,cd,f,mu1,k11,mu2,k2,MHS,path)
       do 1001 i=1,6
        do 1002 j=1,6
         MHS(i,j)=3.0*mu1*MHS(i,j)
1002     continue
1001    continue
       call yieldtest(MHS,sc,sigy1,f,yc,phi1)
       ftest=f+0.001*f
       call Meffective(a,b,c,ad,bd,cd,ftest,mu1,k11,mu2,k2,MHS,path)
       do 1003 i=1,6
        do 1004 j=1,6
         MHS(i,j)=3.0*mu1*MHS(i,j)
1004     continue
1003    continue
       call yieldtest(MHS,sc,sigy1,ftest,yc,phi2)
       dphidf=(phi2-phi1)/(ftest-f)
       H=0.0
       H=-dphidf*(1.0-f)*(nc(1)+nc(2)+nc(3))
     endif
c----- step B -- calculating dphi/dwi1
      if (evolwi.eq..true.) then
       wi1test=wi1+0.001*wi1
       ate=c/wi1test
       call Meffective(ate,b,c,ate,b,c,f,mu1,k11,mu2,k2,
     &                                           MHS,path)
       do 2003 i=1,6
        do 2004 j=1,6
         MHS(i,j)=3.0*mu1*MHS(i,j)
2004     continue
2003    continue
       call yieldtest(MHS,sc,sigy1,f,yc,phi2)
       dphidwi1=(phi2-phi1)/(wi1test-wi1)
      dummy(1)=Amat(3,1)-Amat(1,1)
      dummy(2)=Amat(3,2)-Amat(1,2)
      dummy(3)=Amat(3,3)-Amat(1,3)
      dummy(4)=Amat(3,4)-Amat(1,4)
      dummy(5)=Amat(3,5)-Amat(1,5)
      dummy(6)=Amat(3,6)-Amat(1,6)
      do 400 i=1,6
       dummy5(i)=nc(i)
400   continue
      call rowcolumnprod(dummy,dummy5,dum5)
      H=H-dphidwi1*wi1*dum5
     endif
c----- step C -- calculating dphi/dwi2
      if (evolwi.eq..true.) then
       wi2test=wi2+0.001*wi2
       bte=c/wi2test
       call Meffective(a,bte,c,a,bte,c,f,mu1,k11,mu2,k2,
     &                                           MHS,path)
       do 3003 i=1,6
        do 3004 j=1,6
         MHS(i,j)=3.0*mu1*MHS(i,j)
3004     continue
3003    continue
       call yieldtest(MHS,sc,sigy1,f,yc,phi2)
```

```
         dphidwi2=(phi2-phi1)/(wi2test-wi2)
        do 401 i=1,6
         dummy5(i)=nc(i)
401     continue
        dummy(1)=Amat(3,1)-Amat(2,1)
        dummy(2)=Amat(3,2)-Amat(2,2)
        dummy(3)=Amat(3,3)-Amat(2,3)
        dummy(4)=Amat(3,4)-Amat(2,4)
        dummy(5)=Amat(3,5)-Amat(2,5)
        dummy(6)=Amat(3,6)-Amat(2,6)
        call rowcolumnprod(dummy,dummy5,dum10)
        H=H-dphidwi2*wi2*dum10
       endif
c---------------------------------------------
        L=L+H
c-----------------------------------
        do 450 i=1,6
         dummy5(i)=nc(i)
450     continue
        if(debug.eq..true.) then
        write(15,*)'guess5',L
        call flush(15)
        endif
        call tenmatprod(Ce,strainc,sigec)
        call rowcolumnprod(nc,sigec,dlam)
        dlam=dlam/L
999    return
       end

       subroutine yieldtest(mhs,sig,sigy1,f,yc,yf)
       real*8 mhs(6,6),sig(6),dummy(6),sigy1,f,yf
       logical yc
       yc=.false.
       call tenmatprod(MHS,sig,dummy)
       call rowcolumnprod(sig,dummy,yf)
       yf=yf/((1.0-f)*sigy1)
       yf=yf-sigy1
       if (yf.gt.0.0) then
        yc=.true.
       else
        yc=.false.
       endif
       return
       end

C------ This routine expresses 4th order tensors as 6x6 matrices
c        in the Voigt notation
         subroutine ten2matrix(l1,l)
         real*8 l1(3,3,3,3),l(6,6)
          l(1,1)=l1(1,1,1,1)
          l(1,2)=l1(1,1,2,2)
          l(1,3)=l1(1,1,3,3)
          l(1,4)=dsqrt(2.0d0)*l1(1,1,1,2)
          l(1,5)=dsqrt(2.0d0)*l1(1,1,2,3)
          l(1,6)=dsqrt(2.0d0)*l1(1,1,3,1)
          l(2,1)=l1(2,2,1,1)
          l(2,2)=l1(2,2,2,2)
          l(2,3)=l1(2,2,3,3)
          l(2,4)=dsqrt(2.0d0)*l1(2,2,1,2)
          l(2,5)=dsqrt(2.0d0)*l1(2,2,2,3)
          l(2,6)=dsqrt(2.0d0)*l1(2,2,3,1)
          l(3,1)=l1(3,3,1,1)
          l(3,2)=l1(3,3,2,2)
          l(3,3)=l1(3,3,3,3)
          l(3,4)=dsqrt(2.0d0)*l1(3,3,1,2)
          l(3,5)=dsqrt(2.0d0)*l1(3,3,2,3)
          l(3,6)=dsqrt(2.0d0)*l1(3,3,3,1)
          l(4,1)=dsqrt(2.0d0)*l1(1,2,1,1)
          l(4,2)=dsqrt(2.0d0)*l1(1,2,2,2)
          l(4,3)=dsqrt(2.0d0)*l1(1,2,3,3)
          l(4,4)=(2.0)*l1(1,2,1,2)
          l(4,5)=(2.0)*l1(1,2,2,3)
          l(4,6)=(2.0)*l1(1,2,3,1)
          l(5,1)=dsqrt(2.0d0)*l1(2,3,1,1)
          l(5,2)=dsqrt(2.0d0)*l1(2,3,2,2)
          l(5,3)=dsqrt(2.0d0)*l1(2,3,3,3)
          l(5,4)=(2.0)*l1(2,3,1,2)
          l(5,5)=(2.0)*l1(2,3,2,3)
          l(5,6)=(2.0)*l1(2,3,3,1)
          l(6,1)=dsqrt(2.0d0)*l1(1,3,1,1)
          l(6,2)=dsqrt(2.0d0)*l1(1,3,2,2)
          l(6,3)=dsqrt(2.0d0)*l1(1,3,3,3)
          l(6,4)=(2.0)*l1(1,3,1,2)
          l(6,5)=(2.0)*l1(1,3,2,3)
```

```
        l(6,6)=(2.0)*ll(1,3,3,1)
      return
      end


c------ This is to rotate fourth order tensors
      subroutine rot4order(ll,q,lnew)
      real*8 ll(3,3,3,3),q(3,3),e(9),f(27),d(3)
      real*8 lnew(3,3,3,3)
      integer i,j,k,l
      do 10 i=1,3
       do 11 j=1,3
        do 12 k=1,3
         do 13 l=1,3
          f(1)=q(l,1)*ll(1,1,1,1)+q(l,2)*ll(1,1,1,2)+q(l,3)*ll(1,1,1,3)
          f(2)=q(l,1)*ll(1,1,2,1)+q(l,2)*ll(1,1,2,2)+q(l,3)*ll(1,1,2,3)
          f(3)=q(l,1)*ll(1,1,3,1)+q(l,2)*ll(1,1,3,2)+q(l,3)*ll(1,1,3,3)
          f(4)=q(l,1)*ll(1,2,1,1)+q(l,2)*ll(1,2,1,2)+q(l,3)*ll(1,2,1,3)
          f(5)=q(l,1)*ll(1,2,2,1)+q(l,2)*ll(1,2,2,2)+q(l,3)*ll(1,2,2,3)
          f(6)=q(l,1)*ll(1,2,3,1)+q(l,2)*ll(1,2,3,2)+q(l,3)*ll(1,2,3,3)
          f(7)=q(l,1)*ll(1,3,1,1)+q(l,2)*ll(1,3,1,2)+q(l,3)*ll(1,3,1,3)
          f(8)=q(l,1)*ll(1,3,2,1)+q(l,2)*ll(1,3,2,2)+q(l,3)*ll(1,3,2,3)
          f(9)=q(l,1)*ll(1,3,3,1)+q(l,2)*ll(1,3,3,2)+q(l,3)*ll(1,3,3,3)
          f(10)=q(l,1)*ll(2,1,1,1)+q(l,2)*ll(2,1,1,2)+q(l,3)*ll(2,1,1,3)
          f(11)=q(l,1)*ll(2,1,2,1)+q(l,2)*ll(2,1,2,2)+q(l,3)*ll(2,1,2,3)
          f(12)=q(l,1)*ll(2,1,3,1)+q(l,2)*ll(2,1,3,2)+q(l,3)*ll(2,1,3,3)
          f(13)=q(l,1)*ll(2,2,1,1)+q(l,2)*ll(2,2,1,2)+q(l,3)*ll(2,2,1,3)
          f(14)=q(l,1)*ll(2,2,2,1)+q(l,2)*ll(2,2,2,2)+q(l,3)*ll(2,2,2,3)
          f(15)=q(l,1)*ll(2,2,3,1)+q(l,2)*ll(2,2,3,2)+q(l,3)*ll(2,2,3,3)
          f(16)=q(l,1)*ll(2,3,1,1)+q(l,2)*ll(2,3,1,2)+q(l,3)*ll(2,3,1,3)
          f(17)=q(l,1)*ll(2,3,2,1)+q(l,2)*ll(2,3,2,2)+q(l,3)*ll(2,3,2,3)
          f(18)=q(l,1)*ll(2,3,3,1)+q(l,2)*ll(2,3,3,2)+q(l,3)*ll(2,3,3,3)
          f(19)=q(l,1)*ll(3,1,1,1)+q(l,2)*ll(3,1,1,2)+q(l,3)*ll(3,1,1,3)
          f(20)=q(l,1)*ll(3,1,2,1)+q(l,2)*ll(3,1,2,2)+q(l,3)*ll(3,1,2,3)
          f(21)=q(l,1)*ll(3,1,3,1)+q(l,2)*ll(3,1,3,2)+q(l,3)*ll(3,1,3,3)
          f(22)=q(l,1)*ll(3,2,1,1)+q(l,2)*ll(3,2,1,2)+q(l,3)*ll(3,2,1,3)
          f(23)=q(l,1)*ll(3,2,2,1)+q(l,2)*ll(3,2,2,2)+q(l,3)*ll(3,2,2,3)
          f(24)=q(l,1)*ll(3,2,3,1)+q(l,2)*ll(3,2,3,2)+q(l,3)*ll(3,2,3,3)
          f(25)=q(l,1)*ll(3,3,1,1)+q(l,2)*ll(3,3,1,2)+q(l,3)*ll(3,3,1,3)
          f(26)=q(l,1)*ll(3,3,2,1)+q(l,2)*ll(3,3,2,2)+q(l,3)*ll(3,3,2,3)
          f(27)=q(l,1)*ll(3,3,3,1)+q(l,2)*ll(3,3,3,2)+q(l,3)*ll(3,3,3,3)
          e(1)=q(k,1)*f(1)+q(k,2)*f(2)+q(k,3)*f(3)
          e(2)=q(k,1)*f(4)+q(k,2)*f(5)+q(k,3)*f(6)
          e(3)=q(k,1)*f(7)+q(k,2)*f(8)+q(k,3)*f(9)
          e(4)=q(k,1)*f(10)+q(k,2)*f(11)+q(k,3)*f(12)
          e(5)=q(k,1)*f(13)+q(k,2)*f(14)+q(k,3)*f(15)
          e(6)=q(k,1)*f(16)+q(k,2)*f(17)+q(k,3)*f(18)
          e(7)=q(k,1)*f(19)+q(k,2)*f(20)+q(k,3)*f(21)
          e(8)=q(k,1)*f(22)+q(k,2)*f(23)+q(k,3)*f(24)
          e(9)=q(k,1)*f(25)+q(k,2)*f(26)+q(k,3)*f(27)
          d(1)=q(j,1)*e(1)+q(j,2)*e(2)+q(j,3)*e(3)
          d(2)=q(j,1)*e(4)+q(j,2)*e(5)+q(j,3)*e(6)
          d(3)=q(j,1)*e(7)+q(j,2)*e(8)+q(j,3)*e(9)
          lnew(i,j,k,l)=q(i,1)*d(1)+q(i,2)*d(2)+q(i,3)*d(3)
13       continue
12      continue
11     continue
10    continue
      return
      end


c------ This routine expresses 6x6 matrices back in tensorial(4th order) form.
c       The matrix is in Voigt Notation.
        subroutine mat2tensor(l,ll)
        real*8 l(6,6),ll(3,3,3,3)
        ll(1,1,1,1)=l(1,1)
        ll(1,1,2,2)=l(1,2)
        ll(1,1,3,3)=l(1,3)
        ll(1,1,1,2)=l(1,4)/dsqrt(2.0d0)
        ll(1,1,2,3)=l(1,5)/dsqrt(2.0d0)
        ll(1,1,3,1)=l(1,6)/dsqrt(2.0d0)
        ll(2,2,1,1)=l(2,1)
        ll(2,2,2,2)=l(2,2)
        ll(2,2,3,3)=l(2,3)
        ll(2,2,1,2)=l(2,4)/dsqrt(2.0d0)
        ll(2,2,2,3)=l(2,5)/dsqrt(2.0d0)
        ll(2,2,3,1)=l(2,6)/dsqrt(2.0d0)
        ll(3,3,1,1)=l(3,1)
        ll(3,3,2,2)=l(3,2)
        ll(3,3,3,3)=l(3,3)
        ll(3,3,1,2)=l(3,4)/dsqrt(2.0d0)
        ll(3,3,2,3)=l(3,5)/dsqrt(2.0d0)
```

```
      ll(3,3,3,1)=l(3,6)/dsqrt(2.0d0)
      ll(1,2,1,1)=l(4,1)/dsqrt(2.0d0)
      ll(1,2,2,2)=l(4,2)/dsqrt(2.0d0)
      ll(1,2,3,3)=l(4,3)/dsqrt(2.0d0)
      ll(1,2,1,2)=l(4,4)/2.0
      ll(1,2,2,3)=l(4,5)/2.0
      ll(1,2,3,1)=l(4,6)/2.0
      ll(2,3,1,1)=l(5,1)/dsqrt(2.0d0)
      ll(2,3,2,2)=l(5,2)/dsqrt(2.0d0)
      ll(2,3,3,3)=l(5,3)/dsqrt(2.0d0)
      ll(2,3,1,2)=l(5,4)/2.0
      ll(2,3,2,3)=l(5,5)/2.0
      ll(2,3,3,1)=l(5,6)/2.0
      ll(3,1,1,1)=l(6,1)/dsqrt(2.0d0)
      ll(3,1,2,2)=l(6,2)/dsqrt(2.0d0)
      ll(3,1,3,3)=l(6,3)/dsqrt(2.0d0)
      ll(3,1,1,2)=l(6,4)/2.0
      ll(3,1,2,3)=l(6,5)/2.0
      ll(3,1,3,1)=l(6,6)/2.0
      ll(1,1,2,1)=ll(1,1,1,2)
      ll(1,1,3,2)=ll(1,1,2,3)
      ll(1,1,1,3)=ll(1,1,3,1)
      ll(2,2,2,1)=ll(2,2,1,2)
      ll(2,2,3,2)=ll(2,2,2,3)
      ll(2,2,1,3)=ll(2,2,3,1)
      ll(3,3,2,1)=ll(3,3,1,2)
      ll(3,3,3,2)=ll(3,3,2,3)
      ll(3,3,1,3)=ll(3,3,3,1)
      ll(2,1,1,1)=ll(1,2,1,1)
      ll(2,1,2,2)=ll(1,2,2,2)
      ll(2,1,3,3)=ll(1,2,3,3)
      ll(2,1,1,2)=ll(1,2,1,2)
      ll(2,1,2,3)=ll(1,2,2,3)
      ll(2,1,3,1)=ll(1,2,3,1)
      ll(2,1,2,1)=ll(1,2,1,2)
      ll(2,1,3,2)=ll(1,2,2,3)
      ll(2,1,1,3)=ll(1,2,3,1)
      ll(1,2,2,1)=ll(1,2,1,2)
      ll(1,2,3,2)=ll(1,2,2,3)
      ll(1,2,1,3)=ll(1,2,3,1)
      ll(3,2,1,1)=ll(2,3,1,1)
      ll(3,2,2,2)=ll(2,3,2,2)
      ll(3,2,3,3)=ll(2,3,3,3)
      ll(3,2,1,2)=ll(2,3,1,2)
      ll(3,2,2,3)=ll(2,3,2,3)
      ll(3,2,3,1)=ll(2,3,3,1)
      ll(3,2,2,1)=ll(2,3,1,2)
      ll(3,2,3,2)=ll(2,3,2,3)
      ll(3,2,1,3)=ll(2,3,3,1)
      ll(2,3,2,1)=ll(2,3,1,2)
      ll(2,3,3,2)=ll(2,3,2,3)
      ll(2,3,1,3)=ll(2,3,3,1)
      ll(1,3,1,1)=ll(3,1,1,1)
      ll(1,3,2,2)=ll(3,1,2,2)
      ll(1,3,3,3)=ll(3,1,3,3)
      ll(1,3,1,2)=ll(3,1,1,2)
      ll(1,3,2,3)=ll(3,1,2,3)
      ll(1,3,3,1)=ll(3,1,3,1)
      ll(1,3,2,1)=ll(3,1,1,2)
      ll(1,3,3,2)=ll(3,1,2,3)
      ll(1,3,1,3)=ll(3,1,3,1)
      ll(3,1,2,1)=ll(3,1,1,2)
      ll(3,1,3,2)=ll(3,1,2,3)
      ll(3,1,1,3)=ll(3,1,3,1)
      return
      end

    SUBROUTINE ZBRAK(func,X1,X2,N,XB1,XB2,NB)
    real*8 xb1,xb2,x1,x2,x,dx,fp,fc,func
    integer n,nb,i,nbb
    logical yield, check,neg
    logical zb
    common /mkdata2/yield,check,neg
    common /zb1/zb
    external func
    NBB=NB
    NB=0
    X=X1
    DX=(X2-X1)/N
    FP=func(X)
    if (neg.eq..true.) then
     write(15,*)'returning in ZBRAK'
     call flush(15)
     return
```

```
      endif
      DO 11 I=1,N
        X=X+DX
        FC=func(X)
      if (zb.eq..true.) then
c       write(15,*)'dlam',X, 'f',fc
        endif
      if (neg.eq..true.) then
       write(15,*)'returning in ZBRAK_2'
       call flush(15)
       return
      endif
        IF(FC*FP.LT.0.) THEN
          NB=NB+1
          XB1=X-DX
          XB2=X
c         write(15,*)'info',xb1,xb2,fp,fc
c         call flush(15)
        ENDIF
        FP=FC
        IF(NBB.EQ.NB) then
c       write(15,*)'info2',xb1,xb2,func(xb1),func(xb2)
      RETURN
        endif
11    CONTINUE
      RETURN
      END
```